2011

Real Time Semantic Analysis of Streaming Sensor Data

Harshal Kamlesh Patni
Wright State University

REAL TIME SEMANTIC ANALYSIS OF STREAMING SENSOR DATA

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

By

Harshal Kamlesh Patni
B.E., Mumbai University, 2007

2011

Wright State University

WRIGHT STATE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

December 30, 2011

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Harshal Kamlesh Patni ENTITLED From Real Time Sensor Streams to Real Time Feature Streams BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science

Amit Sheth, Ph.D.
Thesis Director

Mateen Rizki
Chair, Department of Computer Science and Engineering

Committee on
Final Examination

Amit Sheth, Ph.D

Krishnaprasad Thirunarayan, Ph.D

Ramakanth Kavaluru, Ph.D

Andrew Hsu, Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Patni, Harshal Kamlesh. M.S., Department of Computer Science and Engineering, Wright State University, 2011.
Real Time Semantic Analysis of Streaming Sensor Data.

The emergence of dynamic information sources - like social, mobile and sensors, has led to ginormous streams of real time data on the web also called, the era of *Big Data* [1]. Research studies suggest, these dynamic networks have created more data in the last three years than in the entire history of civilization, and this trend will only increase in the coming years [1]. Gigaom[1] article on Big data shows, how the total information generated by these dynamic information sources has completely surpassed the total storage capacity. Thus keeping in mind the problem of ever-increasing data, this thesis focuses on semantically integrating and analyzing multiple, multimodal, heterogeneous streams of weather data with the goal of creating meaningful thematic abstractions in real-time. This is accomplished by implementing an infrastructure for creating and mining thematic abstractions over massive amount of real-time sensor streams. Evaluation section shows 69% data reduction with this approach.

[1] http://gigaom.com/cloud/sensor-networks-top-social-networks-for-big-data-2/

Table of Contents

1. Introduction 1
2. Motivation 7
3. Background and Related Work 9
   - 3.1 Streaming database systems 9
   - 3.2 Stream Reasoning approaches 12
   - 3.3 Temporal RDF 14
   - 3.4 Sensor Data Annotation Framework 15
4. Standards Used 17
   - 4.1 Sensor Data Representation Standards 17
     - 4.1.1 Observation and Measurements (O&M) 18
   - 4.2 Semantic Web 18
     - 4.2.1 Resource Description Framwork (RDF) 19
     - 4.2.2 SPARQL Query Language for RDF 19
     - 4.2.3 Linked Open Data Cloud 20
     - 4.2.4 Ontologies 22
5. Sensor Datasets 25
   - 5.1 Linked Sensor Dataset 26
   - 5.2 Linked Observation Dataset 26
6. Real Time Feature Streams Infrastructure (RTFS) 28
   - 6.1 Sensor Discovery on Linked Data 32
   - 6.2 Semantic Sensor Data Streams 35

6.2.1 Raw Sensor Data Stream 36
6.2.2 O&M Sensor Data Stream 38
6.2.3 Annotated O&M Sensor Data Stream 40
6.2.4 RDF Sensor Data Stream 41
6.3 Feature Streams 44
6.3.1 Integration of Semantic Sensor Data Streams 44
6.3.1.1 System Capability 46
6.3.1.2 Stream Integration 47
6.3.2 Abstraction of Semantic Sensor Data Streams 50
7. Real Time Feature Streams Interface 57
7.1 Begin Search 57
7.1.1 Search Bar 58
7.1.2 State Selection 58
7.2 Feature Selection 58
7.3 Feature Streams View 59
8. Evaluation 61
9. Conclusion and Future Work 65
10. References 66

List of Figures

1. Analysis over single modality data stream....2
2. Feature Stream Computation....4
3. Real Time Feature Streams Architecture (RTFS)....6
4. Linked Sensor Data on Linked Open Data Cloud....22
5. W3C Sensors Ontology....23
6. Sensor Discovery over Linked Data....33
7. Data Conversion Workflow....36
8. Data Stream....36
9. MesoWest Raw Sensor Data Stream....37
10. O&M Sensor Data Stream (focusing on the data values)....40
11. Annotated O&M Sensor Data Stream....41
12. RDF Sensor Data Stream....42
13. Feature Composition....45
14. Feature Composition (RainStorm)....45
15. System Capability....46
16. System Capability (Sensor System "KDAY")....47
17. Stream Integration for feature RainStorm and RainShower....48
18. Integrated Stream Representation....49
19. Integrated Stream for Sensor System KDAY in RDF....50
20. Blizzard Definition....51
21. Flurry Definition....51

22. RainStorm Definition 51

23. RainShower Definition 51

24. Flurry definition encoded in SPARQL 52

25. Feature RDF stream and its relationship 54

26. Feature RDF stream 55

27. Feature Streams Interface (Begin Search) 57

28. Feature Streams Interface (Feature Selection) 59

29. Feature Streams Interface (Feature Streams View) 60

30. Rate of Information Generation 61

31. Storage Evaluation Results 62

List of Tables

1. **Statistics of LinkedObservationData Dataset 27**

Acknowledgement

There are a number of people without whom this thesis might not have been written, and to whom I am greatly indebted.

Foremost, I would like to thank my advisor, Dr. Amit Sheth for his guidance and the opportunity to work and learn from outstanding students at Ohio Center of Excellence in Knowledge-Enabled Computing (Kno.e.sis).

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Krishnaprasad Thirunarayan, and Dr. Ramakanth Kavuluru for their valuable advice and comments.

I owe my most sincere gratitude to Cory Henson, a true friend and a wonderful mentor, who gave me untiring guidance and advice during my research.

Finally, I would like to thank my friends and fellow lab-mates at Kno.e.sis: Ashutosh Jadav, Delroy Cameron, Ashwin Manjunatha, Hemant Purohit, Kalpa Gunaratna, Karthik Gomadam, Meena Nagarajan, Pablo Mendes, Pavan Kapanipati, Pramod Anantraman, Raghava Mutharaju, Sarasi Sarangi, Sujan Perera, Vinh Nguyen, and Wenbo Wang for the stimulating discussions and for the good times we have had working together in the last three years.

Dedicated to my
father Kamlesh for teaching me the importance and value of work,
mother Sunita who has always been there as a strong emotional support,
advisor Dr Amit Sheth for giving me the platform to be where I am today
and friend Cory Henson.

I. INTRODUCTION

Today, over 40 billion sensors estimated to be deployed across the globe are collecting an avalanche of data on everything about and around us. For example a six hour cross-country flight from New York to Los Angeles on a twin-engine Boeing 737 generates a total amount of 240 terabytes of data [1]. On the other hand, with "Internet of Things (IoT)[2] – a layer that bridges the gap between physical and digital world" becoming a reality, massive amount of data can be made available on the web. The sensor revolution combined with IoT would soon result in exceptional awareness of petabytes of sensor data generated by the physical sensors on the web. Thus infrastructures that can process and make sense of massive quantities of data flowing through the web in real-time have gained importance. In other words, with this coming data explosion real-time analytics software must either adapt or die [2]. There has been a lot of work in the database community on analyzing and mining real-time streaming data. Most of the current approaches within the database community provide mathematical summaries (minimum, maximum, average and count) for a single modality stream (like a temperature stream also called lower-level data stream) over time (i.e.

[2] http://en.wikipedia.org/wiki/Internet_of_Things

within a time window) [3, 4]. Figure 1 shows a sample single modality stream "Stream S" with current values in the stream $\{s_1\ s_2\ s_3...s_n\}$ over time $\{t_1\ t_2\ t_3...t_n\}$. "avg" is the average of the current values in the "Stream S" taken over time. The arrow shows the direction of summarization.

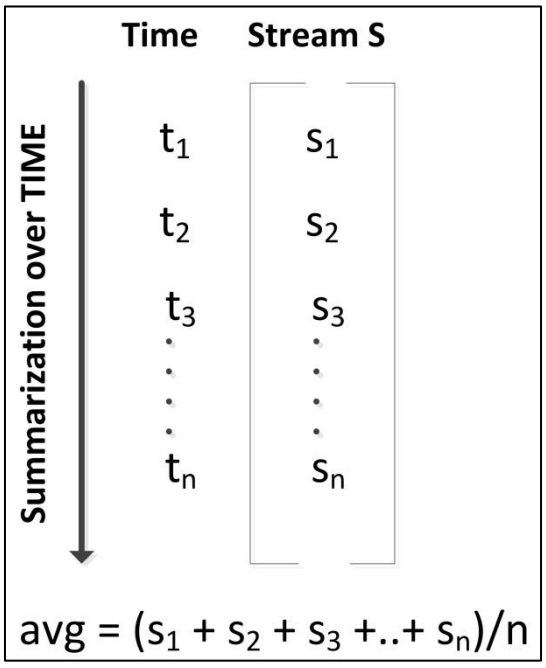


Figure1. Analysis over single modality data stream

Summaries such as (minimum, maximum, average and count) also known as lower-level abstractions are very useful; however they cannot be used to answer questions such as:

1. Is an area X currently detecting a (higher level feature like) blizzard?
2. Find a sequence of higher level features (could be a combination of Blizzard, Flurry, Rainstorm, Rainshower

etc) observed by a weather station over a period of time?

3. When did weather station Y last observe (a higher level feature like) Blizzard?

Danh Le-Phuoc et al. [54] have developed a platform called Linked Sensor Middleware that brings together the live real world sensed data and the Semantic Web. The platform provides wrappers for real time data collection and publishing, interface for data annotation, visualization and a SPARQL end-point for querying unified Linked Stream Data and Linked Data. The data annotation facilitates the integration of sensed data with data from other sources, however such a platform does not help in answering the questions posed above. We think features like Blizzard (also called thematic abstractions) which represent a higher-level of abstraction that humans care about for insight or decision making are much harder to compute since they involve integrating multiple, single modality, heterogeneous lower-level sensor data streams into a higher level feature stream. Such computation for higher level features also requires a thorough knowledge of the weather domain and hence an external knowledge source plays an important role. Figure 2 shows three single modality lower-level data streams “Stream S1, Stream S2, and Stream S3”

and a higher-level feature stream "Feature Stream" over time. "F_s" is computed by integrating the three single modality streams "Stream S1, Stream S2, and Stream S3" and reasoning over thematic dimension using background domain knowledge.

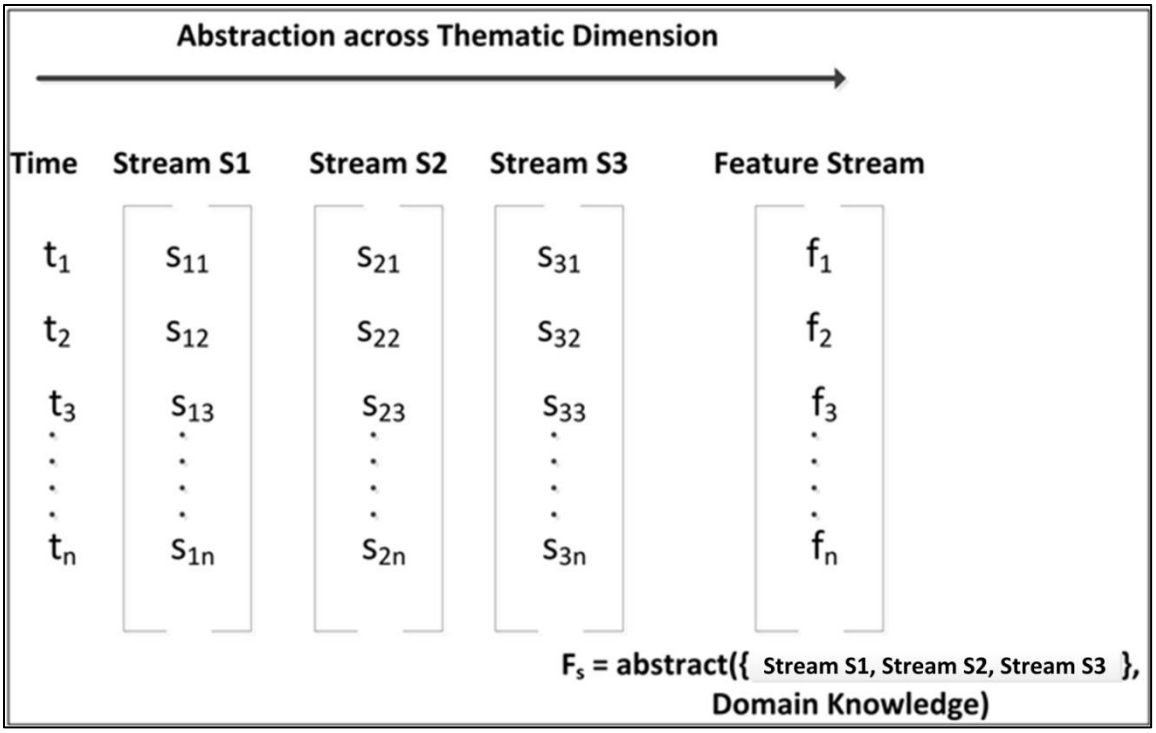


Figure2. Feature Stream Computation

Thus the main research focus of this thesis includes:

- Integration of multiple, multimodal, heterogeneous low-level semantically annotated sensor data streams that compose a feature stream (i.e., stream of abstractions)

- Reasoning over the integrated streams by using domain knowledge and rules to generate feature streams that represent events in the real world. Such abstracted streams require reasoning across the thematic dimension

With the view of answering the questions given above, this thesis provides an infrastructure for creating higher-level feature streams from lower-level data streams. The infrastructure can be divided in three main phases as shown in figure 3. The infrastructure begins with raw sensor data collection. The raw data obtained goes through various transformation phases and is finally annotated using concepts in our ontology to form a stream of RDF triples. This stream of RDF triples is then integrated and reasoned upon constantly using rules within our ontology to detect higher-level features, thus creating feature streams. These feature streams are made publicly accessible by adding them on the Linked Open Data (LOD) Cloud. Section 2 provides a motivating scenario on how creating meaningful abstractions that humans can comprehend makes it easy to analyze and understand events for better decision making. Section 3 provides background information and related work followed by definition of standards (standards from sensor and

semantic web community used within this thesis) in section 4. Section 5 discusses the static sensor datasets added to the Linked Open Data Cloud. The static sensor datasets is not only among the largest dataset on LOD but also the first attempt to add sensor data on LOD. Section 6 focuses on the implemented infrastructure. Section 7 provides details of the interface built over the infrastructure that shows features of interest in real-time. Section 8 evaluates this approach with respect to data storage followed by conclusion and future work.

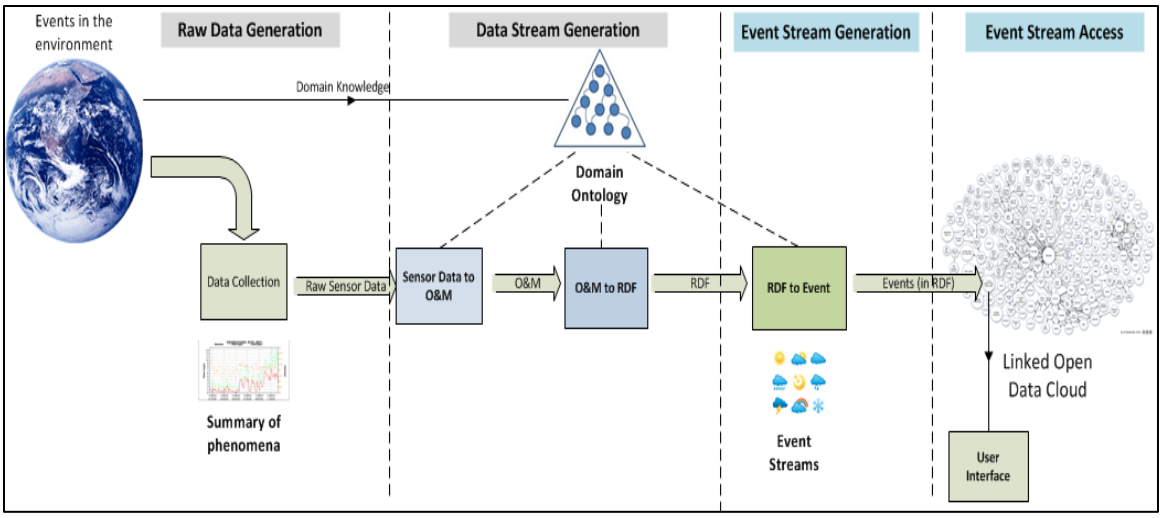


Figure4. Real Time Feature Streams Architecture (RTFS)

II. MOTIVATION

The First North American Blizzard of 2010 was a winter storm and severe weather event that tracked from the U.S. states of California to Arizona through northern Mexico, the American Southwest, and the Midwest, Southeast, and Mid-Atlantic regions. The storm created extensive flooding and landslides in Mexico, as well as historic snowfall totals in the Mid-Atlantic States. The storm stretched from Mexico and New Mexico to New Jersey before moving out to sea, then turning north to impact the Maritime Provinces of Canada. The storm caused deaths in Mexico, New Mexico, Maryland, and Virginia. Washington D.C received heavy snowfall bringing air travel to a halt, suspension of rail service on few routes and limited service on others. Sensors produce huge amount of real-time lower-level data during such events. Storing this massive amount of data is useful; however the scale makes pattern detection and analysis very difficult. Even further the analysis can only be comprehended by meteorologist and weather domain experts [5]. The idea of this thesis is to analyze weather data in real-time, create and store only meaningful higher-level feature streams that humans can easily comprehend for better decision making.

Let us consider the following scenario. Amtrack Operations Center is interested in the sequence of events that resulted in suspension of rail service south of Washington, D.C. To accomplish this task, we would need the following:

1. Search for sensors deployed in the south of Washington D.C where Amtrack provides rail service.
2. Filter the set of sensors capable of detecting the higher level features.
3. Query for the sequence of higher level features (higher level abstractions like Blizzard that humans need for decision making) detected by these sensors before, during and after the natural calamity, and the time at which the rail service was suspended
4. Store only the relevant higher-level features that humans need for decision making.

Such meaningful computation makes it easy for not just weather domain experts but also common man to interpret the sequence of events that lead to suspension of rail service. While these steps may seem easy to answer for an expert, the solution is nontrivial to compute without the use of background knowledge.

III. BACKGROUND AND RELATED WORK

While reasoning tools are year after year scaling up in the classical, time invariant domain of ontological knowledge, there is very little work on reasoning upon rapidly changing information using background knowledge especially where reasoning involves integrating multiple modality and heterogeneous streams with the goal of creating abstraction (higher-level) streams. On the contrary, processing of data streams has been largely investigated and specialized Stream Database Management Systems exist. With this view, this section will provide an overview and comparison with related areas. This work mainly fits in four areas: Streaming database systems [25, 26, 27, 28, 29, and 31], Stream reasoning approaches [38, 39, 40, and 44], temporal RDF [45, 46, 47, and 48] and annotation frameworks for sensor data.

3.1 Streaming Database Systems: Traditional Database Management Systems are built on the concept of persistent data sets that are stored reliably in stable storage and queried/updated several times throughout their lifetime. For several application domains today like automated stock trading, logistic services, business intelligence, sensor networks and social media etc. data arrives and needs to be processed on a continuous (24x7) basis, and

they need data-processing algorithms and systems that work over continuous data streams. [26] This sub-section would focus on discussing the systems built for stream processing and also the languages used for querying data streams.

Data Stream management systems (DSMS) [27] are a new class of data management systems that are specially designed for handling streaming data. The query language has an SQL like syntax especially suited for querying huge volumes of streaming real-time data using time windows called CQL (Continuous query language). Continuous queries were used in the *Tapestry* system [25] for content-based filtering (pattern match) over an append-only database of email and bulletin board messages. The SQL queries did not involve join or time. A user would issue a static query, as though the database was fixed, and Tapestry would convert the static query to continuous query. This simple approach provides guarantees about efficient evaluation and append-only query results. Xyleme [28] is a similar content-based filtering system that monitors pages periodically when fetched and produces an XML report with statistics about the pages fetched with very high throughput and a restricted query language. The *OpenCQ* [29] system

supports continuous queries for monitoring events or update thresholds spread over a wide-area network, e.g., web sites over the Internet. OpenCQ uses a query processing algorithm based on incremental view maintenance. In addition it also provides push-enabled, event driven, content-sensitive information delivery capabilities. NiagaraCQ [30] have similar capabilities as OpenCQ, however NiagaraCQ addresses scalability in number of queries by proposing techniques for grouping continuous queries based on the observation that many web queries share similar structure thus increasing evaluation efficiency. STREAM[31] system supports continuous queries by extending SQL to support stream-oriented primitives such as sliding window operators, timestamp and ordering operators. There are many other streaming database systems such as Aurora [32], Telepgraph [33, 34, 35, 36], however most of these systems handle single streams. Gibbons and Tirthapura [37] executed simple functions such as the number of distinct elements over multiple single modality streams. However these systems are not well suited for applications that use ontologies, where semantic-based event processing and reasoning is required.

3.2 Stream reasoning approaches: There has been a lot of work in the Semantic web community on extending SPARQL with data stream operators. Continuous SPARQL (C-SPARQL) [38] and Streaming SPARQL [39] are languages for continuous query processing and reasoning over streams of RDF data. Both languages extend SPARQL by adding support for window and aggregation operations. Both define time-based and triple-based window operators where upper bound is fixed to current time evaluation, but Streaming SPARQL also supports window creation on historic data. In C-SPARQL, the set of currently valid RDF statements is determined based on the window specification, and classical reasoning on that RDF set is performed as if it were static. On the other hand, Streaming SPARQL approach is built on temporal relational algebra, and the authors provide an algorithm to transform SPARQL queries to that algebra. This thesis focuses on creating a sequence of meaningful higher-level abstractions that humans can easily comprehend using background knowledge. Achieving this goal, requires detection, co-relation, integration of multiple, multimodal RDF streams over time sequence to create abstractions across thematic dimension, which means the temporal order of the triples is very important, in order to view the temporal relatedness

between sequences of higher level events. The temporal relatedness (an event happened before another event) as defined in streaming database systems [41, 42, 43] is required to capture more complex patterns over RDF streaming data [40]. This work uses the approach similar to C-SPARQL as it performs classical reasoning over the current RDF graph as if it was static. However the rules are encoded in SPARQL and executed over the RDF graph. Additionally, in C-SPARQL queries are divided into static and dynamic parts. The static part is evaluated by a RDF triple storage, while a stream processing engine evaluates the dynamic part of the query. This work does not currently support static and dynamic parts. EP-SPARQL Event Processing SPARQL [40] is an extension of SPARQL to support event processing operators. Their focus is more on detection of RDF triples in a specific temporal order (e.g., sequence) as the sequence provides temporal relatedness between events. They propose a unified approach for handling the static and dynamic parts based on logic rules. EP-SPARQL is very similar to this work; however the focus here is on creation of meaningful thematic abstractions and not on extending SPARQL for event processing. Streaming Knowledge Bases [44] is a reasoner dealing with streaming RDF triples and

computation of RDFS closures with respect to ontology. For instance, the reasoner can identify a triple from a stream having a subject that is an instance of a certain class (or any of its subclasses, defined in ontology). In order to speed up stream reasoning, the authors propose to pre-compute all inferences in advance, and to store them in a database. Although this is an interesting approach, however this approach is of little use in real-time reasoning for streaming scenarios and takes considerable amount of time [44], which makes it difficult for this approach to scale.

3.3 Temporal RDF: The authors in [45] provide a notion of incorporating temporal reasoning into RDF, yielding temporal RDF graphs. They provide semantics for temporal RDF graphs and also syntax to incorporate this framework into standard RDF graphs using RDF vocabulary plus temporal labels. The work differs from this thesis in that our aim is to create higher-level abstractions in (near) real time rather than just once posing a query and getting a singular response. This would involve additional reasoning over thematic dimension. The sequence of thematic abstractions created over time, could be used to analyze an event happening in real-time, which requires detection and creation of abstractions

continuously as they happen and hence the data needs to be evaluated constantly. SPARQL-ST [46] is an extension of SPARQL to support complex spatial and temporal queries. The authors provide formal syntax, semantics and an implementation that deals with temporal and spatial data. However, like in [45] the queries need to be triggered and are not continuous. The same argument applies for stSPARQL[47] and T-SPARQL [48].

3.4 Sensor Data Annotation Framework: With avalanche of data being generated by sensors around the globe, the need for analyzing the data to achieve an understanding of our environment has become important. The advent of projects such as OGC[3] Sensor Web enablement (SWE) and W3C Sensor Networks Incubator Group (SSN-XG) [51] has made this information available on the web. However this gives rise to challenges such as discovery, access, search, integration and meaningful use of sensor data on the web. SWE has taken the initial steps to solve these challenges with XML-based languages discussed in the next section. This section focuses on annotation of SWE XML based languages such as O&M. [53] provides a good overview and comparison of semantic annotation languages. Adding semantic annotation to O&M makes the concepts with the

[3] http://www.opengeospatial.org/

O&M explicit, formal and unambiguous. This thesis uses XLINK[4] (XML Linking Language) for annotating concepts within O&M. XLink is a W3C recommendation and outlines methods of describing links between resources in XML documents. Some common attributes of XLink used within this thesis are

- Xlink:type - Every element defining an XLink *must* contain a "type" attribute, which specifies what type of link it is
- xlink:href - The "href" attribute is used to specify the URL of a remote resource, and is mandatory for locator links. In addition to the URL of the remote resource, it may also contain an additional "fragment identifier", which drills down to a specific location within the target document

[4] http://www.w3.org/TR/xlink/

IV. STANDARDS USED

This section explains the standards from sensors and semantic web community used within this thesis.

4.1 Sensor Data Representation Standards

The Open Geospatial Consortium, Inc (OGC)[5] is an international industry consortium of 403 companies, government agencies and universities participating in a consensus process to develop free and openly available interface standards. These standards allow geospatial content and services to be seamlessly integrated "geo-enabling" the web. Developers have made these standards useful by integrating it with all kinds of applications. The Sensor Web Enablement Working Group (SWE)[6] within OGC, are specifying interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the information infrastructure. SWE standards have been well accepted within the sensors community. A complete list of specifications developed and tested by SWE members can be found at the Sensor Web Enablement page [8]. The SWE specifications used in this work can be found below.

[5] http://www.opengeospatial.org/

[6] http://www.opengeospatial.org/projects/groups/sensorweb

- **Observation and Measurements (O&M) -** Observation and Measurements[7] is one of the OGC Sensor Web Enablement (SWE)[8] suites of standards that define an abstract model and an XML schema encoding for sensor observations. O&M is a widely and commonly accepted standard for encoding sensor observation within the sensors community.

4.2 Semantic Web

The Semantic Web[9] is an evolving development of the World Wide Web[10] derived from the World Wide Web consortium (W3C)[11] in which the meaning of information and services on the web is defined, making it possible for the web to understand and satisfy the request of people and machines that use the web content. Tim Berners-Lee coined the term "Semantic Web" and defined it as "a web of data that can be processed directly and indirectly by machines". With this global vision of converting the "web of pages" to "web of data", Semantic Web defines a set of models and technologies; few of the technologies used in this work are discussed below. A complete list of the Semantic Web technologies can be found at the W3C Semantic Web Wiki Page at [9].

[7] http://www.opengeospatial.org/standards/om
[8] http://www.opengeospatial.org/projects/groups/sensorwebdwg
[9] http://www.w3.org/standards/semanticweb/
[10] http://en.wikipedia.org/wiki/World_Wide_Web
[11] http://www.w3.org/

- **Resource Description Framework (RDF) -** Resource Description Framework (RDF)[12] part of the World Wide specifications, is a publishing language within the Semantic Web, specially designed for data. The idea is to make statements about resources on the web in the form of subject-predicate-object known as a "Triple" in the RDF terminology. The subject denotes the resource, the predicate denotes what is being spoken about the resource and the object is the result. The predicate expresses the relationship between the subject and object. A collection of these RDF statements forms a directed labeled graph, a graph describing a resource on the web. An example of graph can be found at [10]. RDF has now come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax formats [11]. It is also a standard model for data interchange on the web.
- **SPARQL Protocol and RDF Query Language -** Since RDF is a directed, labeled graph data format, SPARQL[13] was introduced as a query language. In its usage, SPARQL is a syntactically-SQL-like language for

[12] http://www.w3.org/2001/sw/wiki/RDF
[13] http://www.w3.org/TR/rdf-sparql-query/

querying RDF graphs. The results of SPARQL queries can be results sets or RDF graphs.

- **Linked Open Data Cloud -** The goal of Linked Data is to enable people and organizations to share structured data on the Web as easily as they can share documents today. The term Linked Data was coined by Tim Berners-Lee[14] in his Linked Data Web architecture note. [11] Wikipedia[15] defines Linked Data as "*a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF*". [12] URI's are used to identify any kind of object or concept on the LOD. Resource Description Framework (RDF) is a general-purpose language for data representation on the Web. [13] The basic tenets of Linked Data are described as follows:
  - Use URIs as names for things
  - Use HTTP URIs so that people can look up those names
  - When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)

[14] http://www.w3.org/People/Berners-Lee/
[15] http://www.wikipedia.org/

- Include links to other URIs. So that they can discover more things.[11]

Linked Data is a large and growing collection of interlinked public datasets encoded in RDF that span diverse areas such as: life sciences, nature, science, geography and entertainment. In the sensors domain, sources of geospatial information such as GeoNames[16] and LinkedGeoData[17] are of particular importance. The GeoNames geographical dataset contains over eight million geographical names and consists of 7 million unique features including 2.6 million populated places and 2.8 million alternate names. In the next section, we introduce two sensor datasets built as a part of the Semantic Sensor Web Project [15] at Kno.e.sis [16], *LinkedSensorData* and *LinkedObservationData*. The sensor datasets contribute ~1.7 billion triples to the LOD which makes it among the largest datasets on the Linked Open Data Cloud. This is also the first attempt to add sensor data to the Linked Open Data Cloud. Figure 4 shows the Linked Sensor Data (Kno.e.sis)[18] as a part of the Linked Open Data Cloud.

[16] http://www.geonames.org/
[17] http://linkedgeodata.org/About
[18] http://thedatahub.org/dataset/knoesis-linked-sensor-data

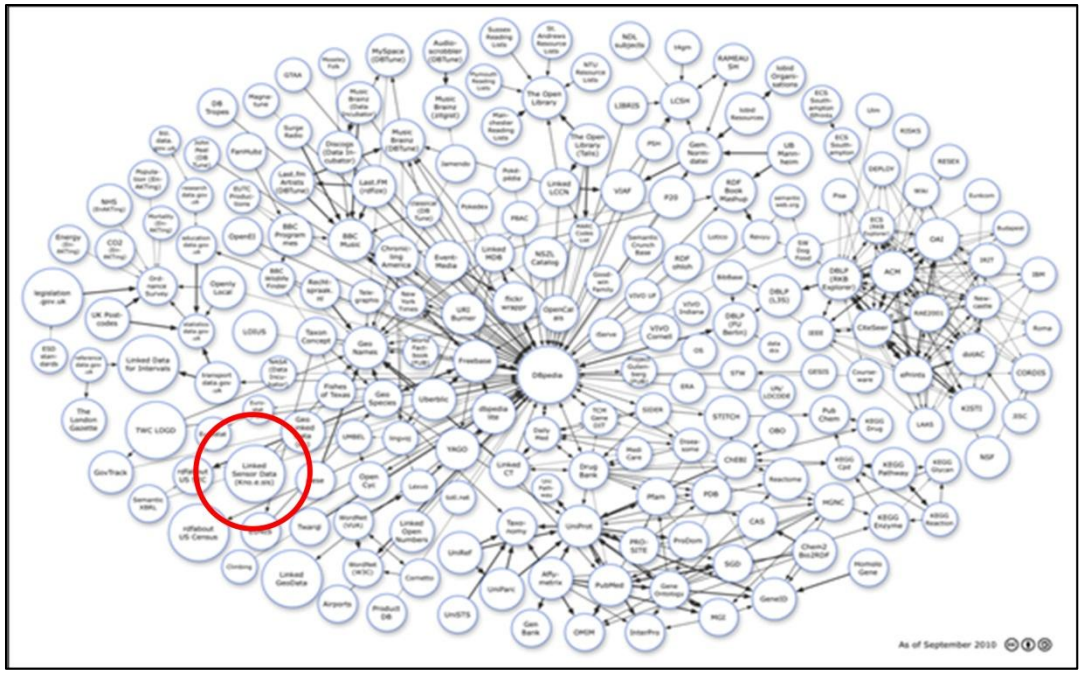


Figure4. Linked Sensor Data on Linked Open Data Cloud

- **Ontologies -** In computer science and information science, ontology is a formal representation of knowledge by a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may be used to describe the domain [17]. This thesis uses the W3C Semantic Sensor Network Ontology[19] [51]. W3C has developed a formal OWL-DL ontology, the Semantic Sensor Network Ontology (SSN), for modeling sensor devices (and their capabilities), systems, and processes. The development was informed by a thorough review of previous sensor ontologies by [52], and drawing on earlier

[19] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

vocabularies like those of the OGC SWE (SensorML and O&M). Figure 5 provides an overview of the main concepts and structure of the ontology.

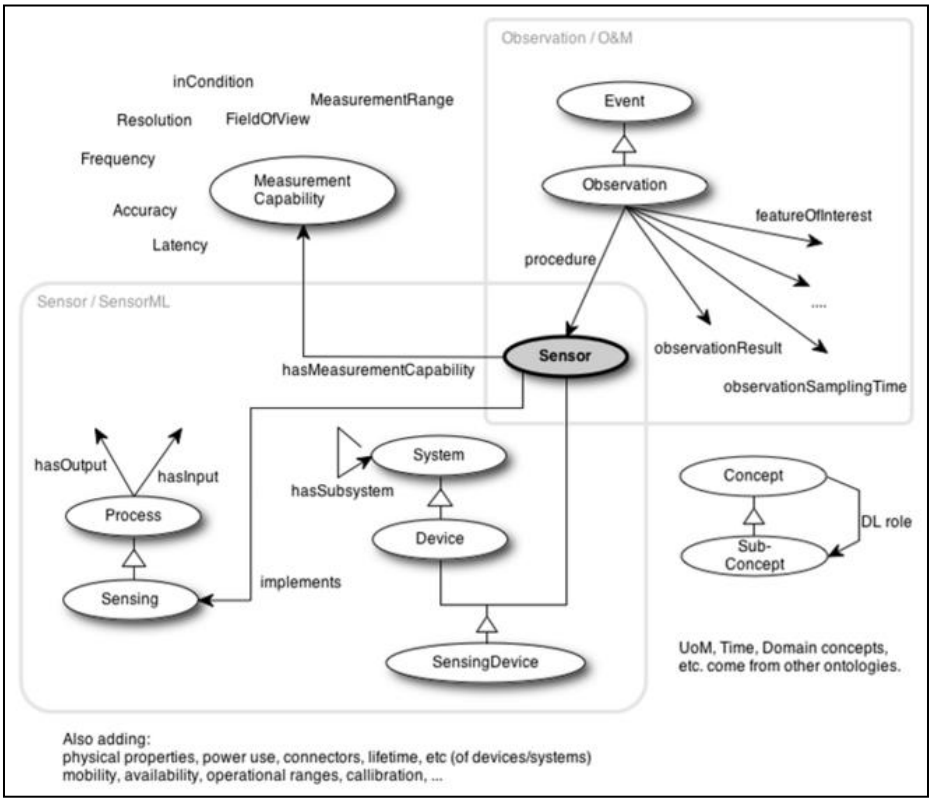


Figure5. W3C Sensors Ontology

The SSN ontology is based around concepts of systems, processes, and observations. It supports the description of the physical and processing structure of sensors. This thesis uses the Observation O&M component of this ontology. Here an observation (Observation) is defined as an act of observing a property or phenomenon, with the goal of producing an estimate of the value of the property, and a feature (Feature) is defined as an abstraction of real world

phenomenon. This thesis focuses on creating these higher-level features from lower-level data in real-time. The major properties of an observation include feature of interest (featureOfInterest), sampling time (observationSamplingTime), result (observationResult), and procedure (procedure). Often these properties can be complex entities that may be defined in an external document. For example, featureOfInterest could refer to any real-world entity such as a coverage region, vehicle, or weather-storm, and procedure often refers to a sensor or system of sensors, and properties are described as relationships of an observation.

V. SENSOR DATASETS

This section focuses on the two sensor datasets LinkedSensorData and LinkedObservationData. As previously stated, this was the first attempt on adding sensor data to the LOD as well as among the largest datasets contributing ~1.7 billion triples. Each dataset is described in detail below.

5.1 Linked Sensor Data - LinkedSensorData is an RDF dataset containing expressive descriptions of ~20,000 weather stations in the United States. The dataset confines to the W3C sensors ontology. The data originated at MesoWest [4], a project within the Department of Meteorology at the University of Utah[20] that has been aggregating weather data since 2002. [13] On average, there are five sensors per weather station measuring phenomena such as temperature, visibility, precipitation, pressure, wind speed, humidity, etc. In addition to location attributes such as latitude, longitude, and elevation, there are links to locations in Geonames near the weather station. The distance from the Geonames location to the weather station is also provided. The data set also contains links to the most current observation for each weather station also

[20] http://www.utah.edu/

provided by MesoWest. This sensors description dataset is now part of the LOD. More information about the datasets can be found at [49]

5.2 Linked Observation Data - LinkedObservationData is an RDF dataset containing expressive descriptions of hurricanes and blizzard observations in the United States. The data again originated at MesoWest. The observations collected include measurements of phenomena such as temperature, visibility, precipitation, pressure, wind speed, humidity, etc. The weather station's observations also include the unit of measurement for each of these phenomena as well as the time instant at which the measurements were taken. The dataset includes observations within the entire United States during the time periods that several major storms were active -- including Hurricane Katrina, Ike, Bill, Bertha, Wilma, Charley, Gustav, and a major blizzard in Nevada in 2002. These observations are generated by weather stations described in the LinkedSensorData dataset introduced above. Currently, this dataset contains more than a billion triples. The RDF dataset for each of the above storms is available for download in gzip format at

[18]. The statistics for these storms can be found in the Table 1 below.

Name	Storm Type	Date	No Of Triples	No Of Observation
All			1,730,284,735	159,460,500
Bill	Hurricane	August 17 - August 22, 2009	231,021,108	21,272,790
Ike	Hurricane	September 1 - September 13, 2008	374,094,660	34,430,964
Gustav	Hurricane	September 1 - September 13, 2008	258,378,511	23,792,818
Bertha	Hurricane	July 6 - July 17, 2008	278,235,734	25,762,568
Wilma	Hurricane	October 17 - October 23, 2005	171,854,686	15,797,852
Katrina	Hurricane	August 23 - August 30, 2005	203,386,049	18,832,041
Charley	Hurricane	August 9 - August 15, 2004	101,956,760	9,333,676
	Blizzard	April 1 - April 6, 2003	111,357,227	10,237,791

Table.1. Statistics for LinkedObservationData Dataset

VI. Real-Time Feature Streams Infrastructure (RTFS)

This section focuses on the infrastructure for creating thematic abstractions over streaming sensor data in real-time. Before describing the infrastructure in detail, let us have a look at the research issues that are addressed while building the infrastructure for creating abstractions in real-time. The research issues include:

- Integration of multiple, multimodal, heterogeneous low-level semantically annotated sensor data streams that compose a feature stream (i.e., stream of abstractions)
- Reasoning over the integrated streams by using background knowledge and rules to generate feature streams that represent events in the real world. Such abstracted streams require reasoning across the thematic dimension

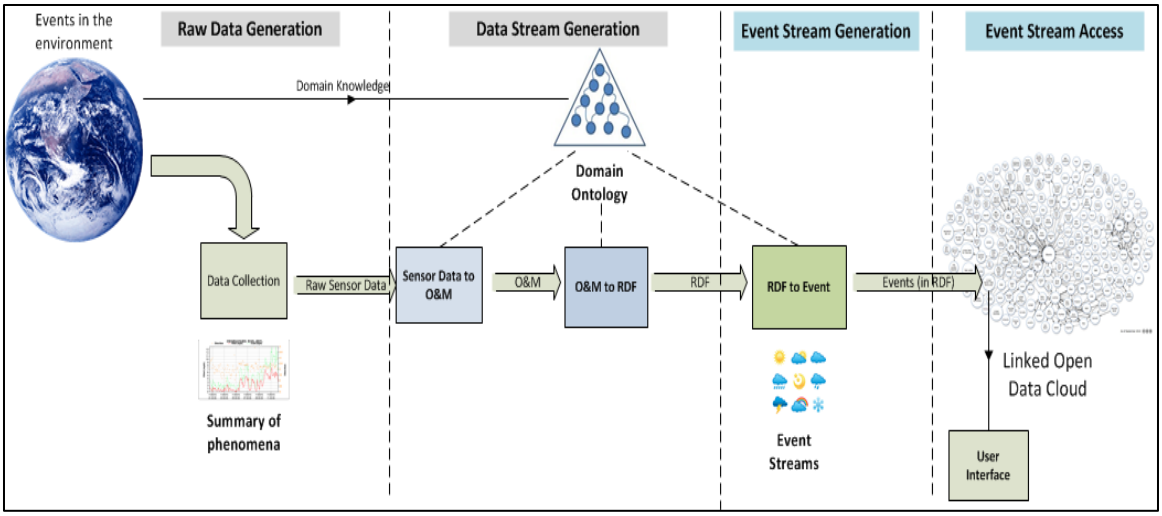


Figure3. Feature Streams Architecture

With the view of addressing these research issues; Figure 3 (repeated again here for ease) shows architecture for Feature Streams which is divided into three parts based on the functionality as shown below:

- **Semantic Sensor Data Streams:** Conversion of raw sensor data streams to RDF sensor data stream. This conversion step forms the pillar for achieving the research issues stated above
- **Integration of Semantic Sensor Data Streams:** Discover and integrate streams that compose a feature stream
- **Abstraction of Semantic Sensor Data Streams:** Reason over the integrated stream across the thematic dimension using background knowledge to generate

feature streams. The feature streams are made openly accessible by adding to the Linked Open Data Cloud.

In order to make the explanation interesting, let us consider a scenario and see how the steps given above, help solving the scenario below.

Scenario – "Find a sequence of weather events that are currently being observed near Dayton James Cox Airport?"

The scenario above can be divided into 3 dimensions listed below [19]:

- Space: Spatial dimension describes the location where the event is occurring (Dayton James Cox Airport)
- Time: Temporal dimension describes the time of the event (current/happening right now)
- Theme: Thematic dimension describes what are we looking for (weather events like Blizzard, Flurry etc)

In the next few pages, the scenario is broken in few parts and the technology used to solve each part will be explained in detail.

Scenario Part 1

Problem: *Find sensors near Dayton James Coz Airport?*

Technology: *Sensor Discovery on Linked Data [20]*

Dimension: *Spatial Dimension*

6.1 Sensor Discovery on Linked Data

This section is not a part of this thesis and hence would not be explained in technical depth. A quick overview on this work would be provided as it helps solving the scenario. Since huge numbers of sensors collect data about our environment, finding relevant sensors on the web is a non-trivial challenge. Although one can use sensor co-ordinates to search for sensors, however this process is very unintuitive for anyone who is not a part of the sensors community. Pschorr et al. [20] approach this problem of discovering sensors by providing a standard service interface over Linked Data [20]. The standard service interface is a semantically annotated extension of OGC SWE's Sensor Observations Service (SOS)[21] which is used to execute queries over Linked Data as a sensor registry. The authors leverage the Geonames [21] dataset on LOD. As described before, Geonames contains expressive descriptions of spatial data and named locations on the web. The sensor descriptions datasets Linked Sensor Data, described in section 5.2 was linked to Geonames to enhance the sensor descriptions with named locations. Relating sensor descriptions to nearby locations defined within Geonames allows intuitive sensor discovery queries using named

[21] http://www.opengeospatial.org/standards/sos

locations like "Dayton James Cox Airport". Figure 6 provides a snapshot of the interface built to search for sensors with named location. The snapshot uses "Dayton James Cox Airport" as an example. The result contains a sensor system "KDAY" near "Dayton James Cox Airport". The result provides the latitude, longitude, properties that system can observe, link to the location from Geonames and also the current observations from MesoWest [4].

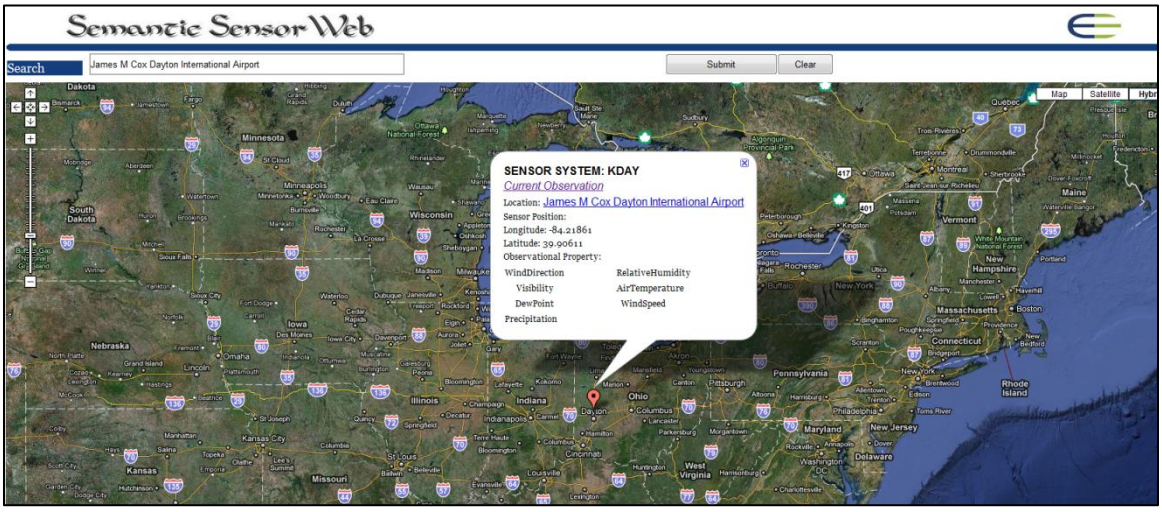


Figure6. Sensor Discovery over Linked Data

Scenario Part 2

Problem: *Extract real-time data for sensors near Dayton James Cox Airport?*

Technology: *Semantic Sensor Data Streams*

Dimension: *Temporal Dimension*

6.2 Semantic Sensor Data Streams

In scenario part 1, we searched for the sensors near "Dayton James Cox Airport". This section focuses on extracting real-time data for the sensor system "KDAY" found in scenario part 1, and converts the raw data stream to an RDF data Stream. As discussed before this section is also the building block for the integration, analysis and generation of feature streams discussed in the section 6.3 and 6.4. The process begins at MesoWest a project within the Department of Meteorology at the University of Utah that has been aggregating weather data since 2002. However it could be replaced with any source that provides access to sensor observations. A few sources that provide access to weather observations are Google [23], NOAA [24] etc. Figure 7 shows the whole data conversion process in the form of a data conversion workflow. The RDF sensor data stream generated is used as the input for section 6.3 (semantic integration of sensor data stream) and 6.4 (reasoning over integrated semantic sensor data stream). The next few sub-sections would discuss the data conversion process in detail.

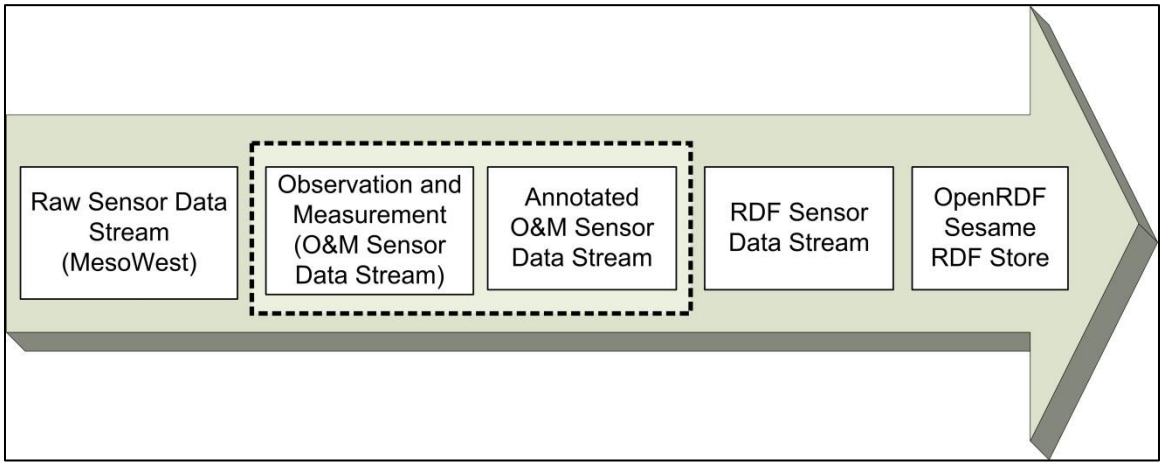


Figure7. Data Conversion Workflow

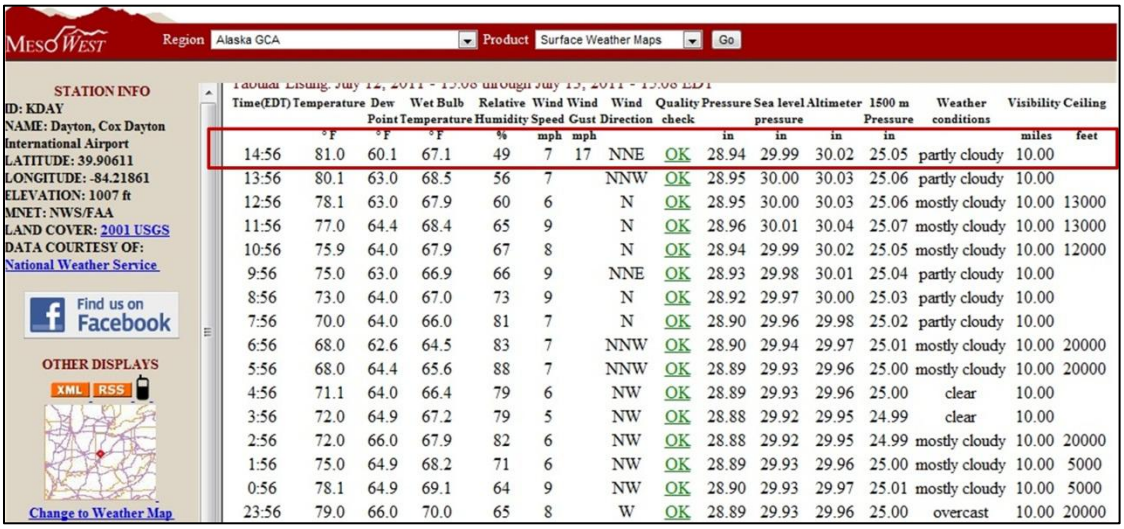
- **Raw Sensor Data Stream -** Data stream is a sequence of time separated data values. Figure 8 shows a data stream "Data Stream" with values {d_1, d_2, d_3..d_n} over time points {t_1, t_2, t_3..t_n}.

Time	Data Stream
t_1	d_1
t_2	d_2
t_3	d_3
⋮	⋮
t_n	d_n

Figure8. Data Stream

A raw sensor data stream is a sequence of time separated, lower-level (numerical) sensor observation

readings. Figure 9 provides a snapshot of a raw sensor data stream for sensor system "KDAY" containing current numerical observation values from MesoWest. The red box shows numerical values for Temperature, Wind Speed, Precipitation etc. at time instant 2:56pm. The image contains a sequence of these values over a period of time from 12:56am to 2:56pm which forms a stream of raw sensor data values.



Time(EDT)	Temperature °F	Dew Point °F	Wet Bulb Temperature °F	Relative Humidity %	Wind Speed mph	Wind Gust mph	Wind Direction	Quality check	Pressure in	Sea level pressure in	Altimeter in	1500 m Pressure in	Weather conditions	Visibility miles	Ceiling feet
14:56	81.0	60.1	67.1	49	7	17	NNE	OK	28.94	29.99	30.02	25.05	partly cloudy	10.00	
13:56	80.1	63.0	68.5	56	7		NNW	OK	28.95	30.00	30.03	25.06	partly cloudy	10.00	
12:56	78.1	63.0	67.9	60	6		N	OK	28.95	30.00	30.03	25.06	mostly cloudy	10.00	13000
11:56	77.0	64.4	68.4	65	9		N	OK	28.96	30.01	30.04	25.07	mostly cloudy	10.00	13000
10:56	75.9	64.0	67.9	67	8		N	OK	28.94	29.99	30.02	25.05	mostly cloudy	10.00	12000
9:56	75.0	63.0	66.9	66	9		NNE	OK	28.93	29.98	30.01	25.04	partly cloudy	10.00	
8:56	73.0	64.0	67.0	73	9		N	OK	28.92	29.97	30.00	25.03	partly cloudy	10.00	
7:56	70.0	64.0	66.0	81	7		N	OK	28.90	29.96	29.98	25.02	partly cloudy	10.00	
6:56	68.0	62.6	64.5	83	7		NNW	OK	28.90	29.94	29.97	25.01	mostly cloudy	10.00	20000
5:56	68.0	64.4	65.6	88	7		NNW	OK	28.89	29.93	29.96	25.00	mostly cloudy	10.00	20000
4:56	71.1	64.0	66.4	79	6		NW	OK	28.89	29.93	29.96	25.00	clear	10.00	
3:56	72.0	64.9	67.2	79	5		NW	OK	28.88	29.92	29.95	24.99	clear	10.00	
2:56	72.0	66.0	67.9	82	6		NW	OK	28.88	29.92	29.95	24.99	mostly cloudy	10.00	20000
1:56	75.0	64.9	68.2	71	6		NW	OK	28.89	29.93	29.96	25.00	mostly cloudy	10.00	5000
0:56	78.1	64.9	69.1	64	9		NW	OK	28.90	29.93	29.97	25.01	mostly cloudy	10.00	5000
23:56	79.0	66.0	70.0	65	8		W	OK	28.89	29.93	29.96	25.00	overcast	10.00	20000

Figure9. MesoWest Raw Sensor Data Stream

The raw sensor data stream is obtained by querying MesoWest for sensor observational data and parsing the result. MesoWest provides a service to access real-time data that takes station ID, current date, and time as parameters. The result is an HTML page (figure 9) containing observations made by the weather station for the past one hour. Since MesoWest does not push

new data this process is continued every hour. The resulting HTML page is then parsed to extract the sensor observational data.

- **O&M Sensor Data Stream -** O&M (Observation and Measurements) is an OGC Sensor Web Enablement suite of standards for encoding lower-level sensor observations. O&M is a widely accepted standard within the sensors community and hence forms an important step within the conversion process. An O&M Sensor Data Stream is a time separated sequence of sensor observation readings (numerical readings) encoded in O&M. Figure 11 shows O&M that encodes observations for Temperature, Wind Speed and Precipitation. The "name" field within the O&M provides the name of the observed property (AirTemperature, Wind Speed etc.). The "swe:Quantity" field provides the URI to the concept in the source ontology. In figure 11, the URI for the concepts within the "swe:Quantity" field is obtained from SWEET ontology [22]. SWEET ontology (Semantic Web for Earth and Environmental terminology) was built by NASA's (National Aeronautics and Space Administration)[22] Jet Propulsion Laboratory[23] with the view of providing

[22] http://www.nasa.gov
[23] http://www.jpl.nasa.gov/

a common semantic framework for Earth Science initiatives. “swe:uom” field provides the URI for the Unit of Measurement obtained from OGC. The “swe:values” field shows the numerical values for observations over time encoded in O&M. The order of the observations in the “values” field follows the order of “name” field. It begins with the timestamp, followed by Temperature, Wind Speed and Precipitation values. Timestamps are separated by “@@”. The red box in figure 10 shows the observations values for Temperature “24.1”, Wind Speed “19.1” and Precipitation “snow” at time instant “2011-2-28T10:56:00-07:00”. Since the O&M contains concepts from various ontologies, we would create an annotated O&M in next sub-section, where the concepts are mapped to the W3C sensor ontology. The appendix section contains a complete example of weather station and its observations encoded in O&M.

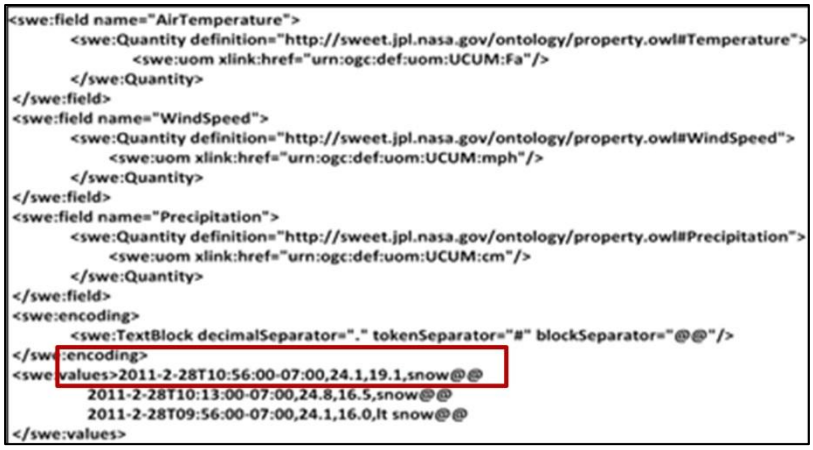


Figure10. O&M Sensor Data Stream (focusing on the data values)

Annotated O&M Sensor Data Stream - An annotated O&M Sensor Data Stream is a time separated sequence of sensor observation readings (numerical readings) encoded in O&M annotated with concepts within the W3C sensor ontology. As discussed before in section 4.2 W3C sensors ontology is a standard ontology that describes concept within the sensors domain. Annotating the O&M with W3C's ontology concepts results in more descriptive, standardized and semantically enriched O&M. Figure 11 annotates the O&M obtained from figure 10. The "swe:Quantity" field in the annotated O&M contains the URI to concepts in the W3C sensors ontology (highlighted in red).

```
<swe:field name="AirTemperature">
        <swe:Quantity definition="http://purl.oclc.org/NET/ssnx/ssn#AirTemperature">
                <swe:uom xlink:href="fahrenheit"/>
        </swe:Quantity>
</swe:field>
<swe:field name="WindSpeed">
        <swe:Quantity definition="http://purl.oclc.org/NET/ssnx/ssn#WindSpeed">
            <swe:uom xlink:href="milesPerHour"/>
        </swe:Quantity>
</swe:field>
<swe:field name="Precipitation">
        <swe:Quantity definition="http://purl.oclc.org/NET/ssnx/ssn#Precipitation">
            <swe:uom xlink:href="centimeters"/>
        </swe:Quantity>
</swe:field>
<swe:encoding>
        <swe:TextBlock decimalSeparator="." tokenSeparator=",&#9;" blockSeparator="@@"/>
</swe:encoding>
<swe:values>2011-2-28T10:56:00-07:00,24.1,19.1,snow@@
            2011-2-28T10:13:00-07:00,24.8,16.5,snow@@
            2011-2-28T09:56:00-07:00,24.1,16.0,lt snow@@
</swe:values>
```

Figure11. Annotated O&M Sensor Data Stream with W3C SSN concepts

- **RDF Sensor Data Stream -** An RDF Sensor Data Stream is a time separated sequence of sensor observation readings (numerical readings) encoded in RDF using the W3C sensor ontology. Since both O&M and RDF have an XML like syntax, we generated an XSLT[24] to convert annotated O&M to RDF. XSLT is a language for transforming XML documents into other XML documents. The RDF data generated in this phase is stored in Sesame OpenRDF[25] store. Figure 12 encodes the Temperature observation value "24.1" at time "2011-02-28T10:56:00" as an RDF stream. Although the RDF was

[24] XSLT, http://www.w3.org/TR/xslt
[25] http://www.openrdf.org/

generated in RDF/XML format, figure 12 shows N-Triple format for better readability.

```
sens-obs:Observation_AirTemperature_KDAY_2011_02_28_10_56_00
    a      weather:TemperatureObservation ;
    om-owl:observedProperty
          weather:_AirTemperature ;
    om-owl:procedure sens-obs:System_KDAY ;
    om-owl:result sens-
obs:MeasureData_AirTemperature_KDAY_2011_02_28_10_56_00 ;
    om-owl:samplingTime sens-obs:Instant_2011_02_28_10_56_00 .

sens-obs:MeasureData_AirTemperature_KDAY_2011_02_28_10_56_00
    a      om-owl:MeasureData ;
    om-owl:floatValue "24.1" ;
    om-owl:uom weather:fahrenheit .

sens-obs:Instant_2011_02_28_10_56_00
    a      owl-time:Instant ;
    owl-time:inXSDDateTime
          "2011-02-28T10:56:00"
```

Figure12. RDF Sensor Data Stream

The generation of RDF sensor data stream is the last step of data conversion process. The RDF Sensor Data Stream would now be used to search for higher-level events like Blizzard, Flurry etc. in the next few sections.

Scenario Part 3

Problem: *What events are being observed near Dayton James Cox Airport currently?*

Technology: *Feature Streams*

Dimension: *Thematic Dimension*

6.3 Feature Streams

Scenario part 2, focused on extracting raw sensor data streams in real-time for sensor system KDAY and conversion to RDF sensor data stream. This section focuses on integration, analysis and reasoning over RDF data streams generated in the previous section to search for higher-level features like Blizzard, Flurry, RainStorm, RainShower etc. The weather ontology mapped to W3C SSN ontology is used as background knowledge for reasoning over RDF data streams. The section is divided into two sub-sections below:

6.3.1 Integration of Semantic Sensor Data Streams

This section focuses on semantically integrating the streams that compose a feature. Before we semantically integrate streams that compose a feature, let us define the relationship between a feature and its property. Figure 14 shows a "feature" and "property" relationship. The "hasProperty" relationship defines what a feature is composed of. The feature f_1 in figure 13 is composed of properties $\{p_1, p_3, p_n\}$, while feature f_2 is composed of properties $\{p_1, p_2, p_3\}$. Figure 14 applies this definition to the weather domain.

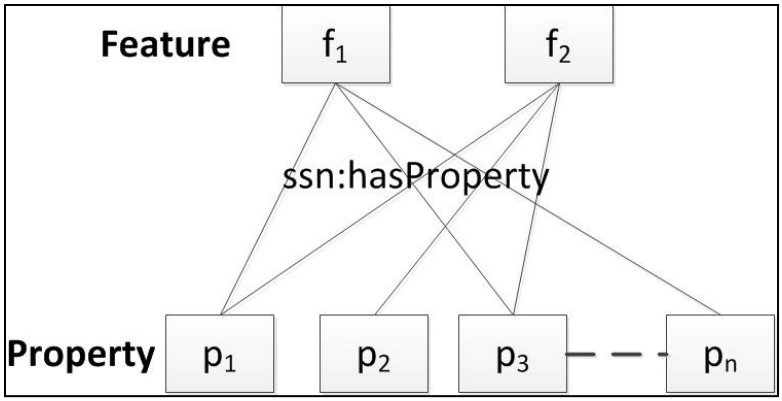


Figure13. Feature Composition

As shown in figure 14, a RainStorm feature is composed of properties {High WindSpeed, Rain Precipitation and Non Freezing Temperature}. In other words, for a higher-level feature RainStorm to be detected, the lower-level weather conditions must contain high WindSpeed, Rain Precipitation and Temperature above freezing conditions

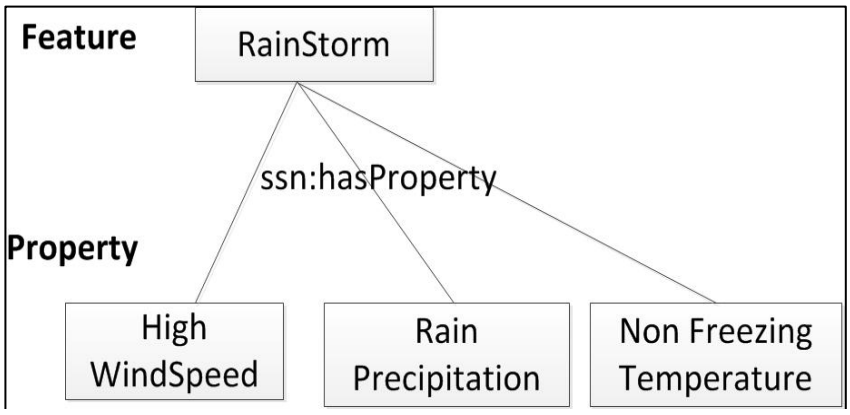


Figure14. Feature Composition (RainStorm)

For better explanation the section is further divided into the following sub-sections. The sub-sections provide explanation based on 2 key functionalities given below.

6.3.1.1 System Capability

The previous section focused on feature composition. This section focuses on sensor systems and its observing capability. As shown in figure 15 a system is composed of various types of sub-systems (also called sensors). In this example, sys_1 is composed of sub-systems {s_1, s_3, s_4}. And each sub-system (sensor) is capable of observing a unique property.

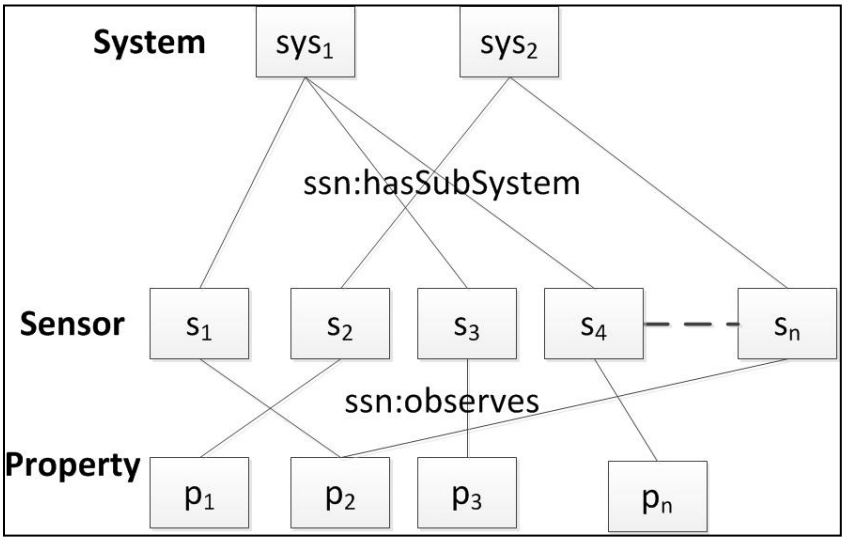


Figure15. System Capability

Applying this to the weather domain, figure 16 shows a system KDAY that has sub-systems {Temperature Sensor, Wind Sensor, Precipitation Sensor, Pressure Sensor} and hence can observe {Temperature, WindSpeed, Rain and Pressure} properties. This example is simplified for ease of explanation. In real-world a system is

composed of on average 7-8 sensors capable of observing 7-8 unique properties.

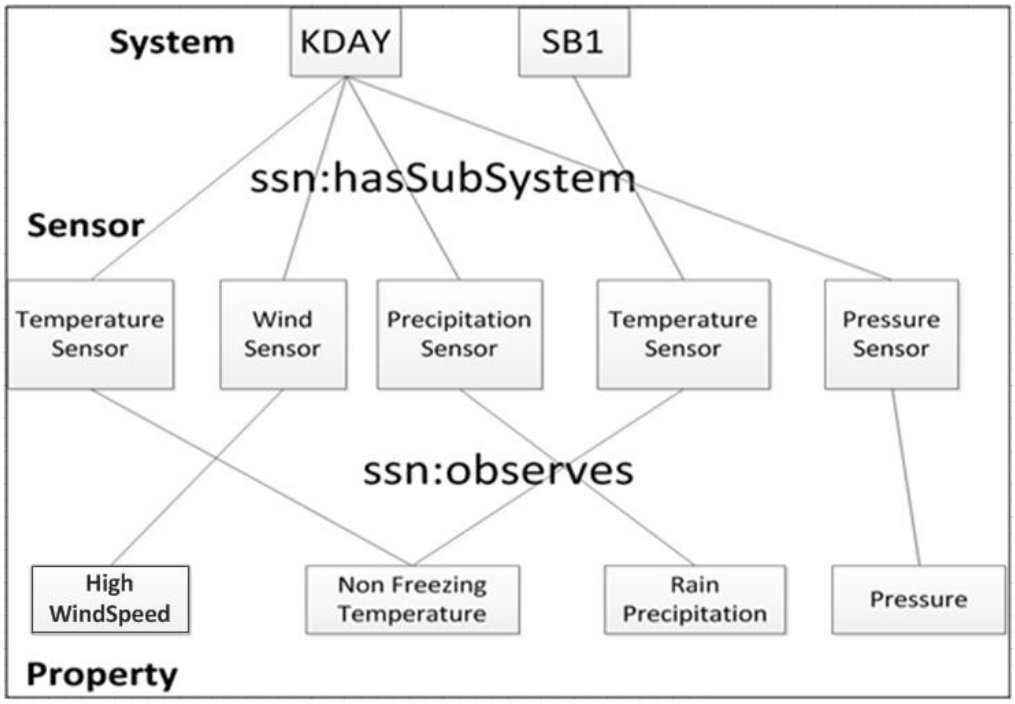


Figure16. System Capability (Sensor System "KDAY")

6.3.1.2 Integrate streams that compose a feature

RainStorm in figure 14 is composed of {High WindSpeed, Non Freezing Temperature and Rain Precipitation} properties, while system KDAY in figure 16 is capable of observing {High WindSpeed, Non Freezing Temperature, Rain Precipitation and Pressure} which includes the properties that RainStorm feature is composed of. Thus integrating three streams {High WindSpeed, Non Freezing Temperature and Rain Precipitation} in figure 17, we see system KDAY is

capable of detecting a feature RainStorm. Since RainShower feature is also composed of the same properties as RainStorm, system KDAY can also detect RainShower.

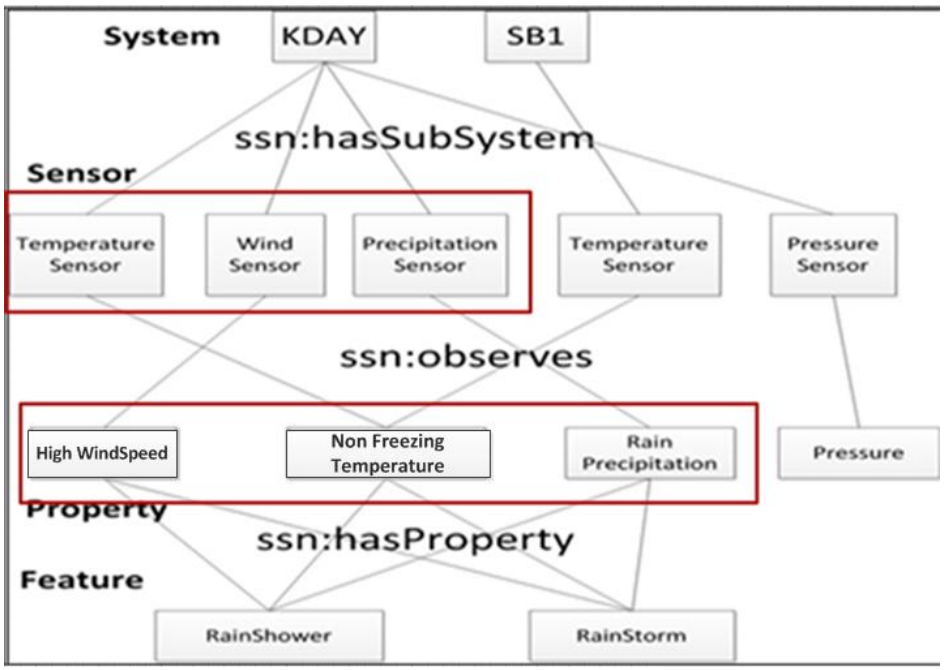


Figure17. Stream Integration for feature RainStorm and RainShower

Figure 18 shows the integrated stream, by integrating observation values from three lower-level streams Non Freezing Temperature, High WindSpeed and Rain Precipitation over time. The next section uses background knowledge encoded in weather ontology to analyze these integrated streams at time $\{t_1\ t_2\ t_3 \ldots t_n\}$ to search for higher level features. Analyzing

integrated streams over time would create a stream of higher-level features (feature stream).

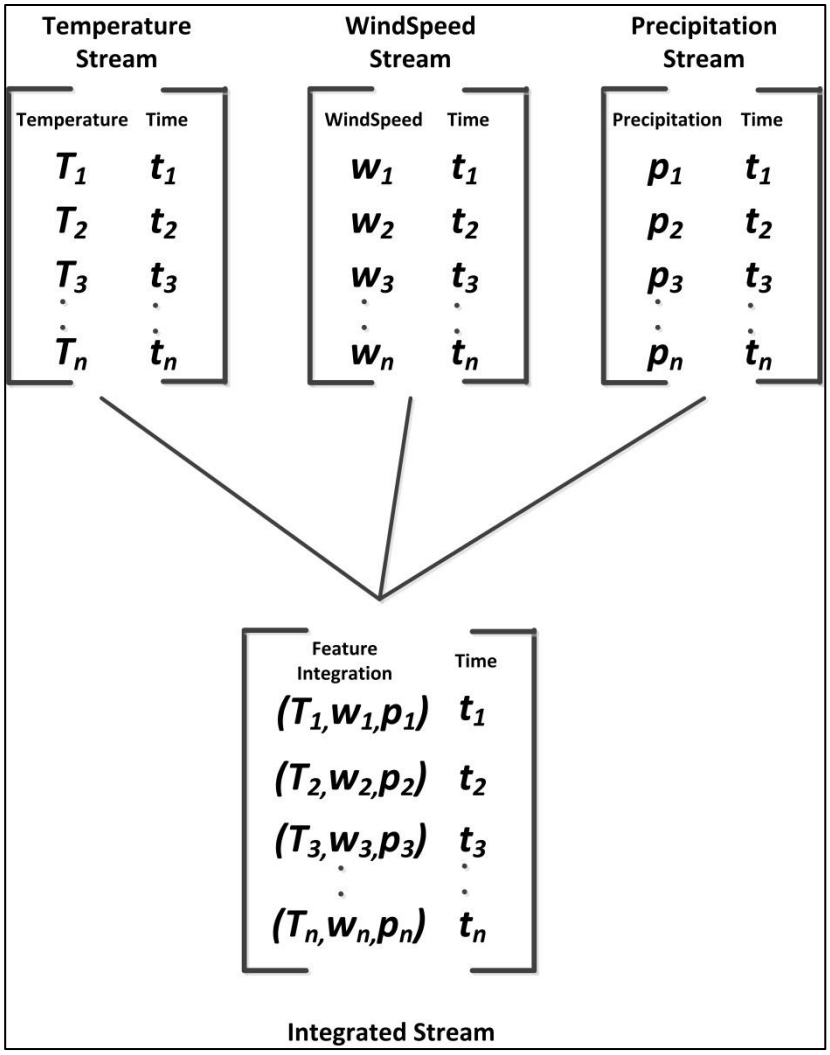


Figure18. Integrated Stream Representation

Figure 19 shows the integrated stream for sensor system KDAY in RDF. The integrated streams contain Temperature, WindSpeed and Precipitation values for sensor system KDAY encoded in RDF at time "2011-02-28T10:56:00".

```
sens-obs:Observation_AirTemperature_KDAY_2011_02_28_10_56_00
    a      weather:TemperatureObservation ;
    om-owl:observedProperty
          weather:_AirTemperature ;
    om-owl:procedure sens-obs:System_KDAY ;
    om-owl:result sens-obs:MeasureData_AirTemperature_KDAY_2011_02_28_10_56_00 ;
    om-owl:samplingTime sens-obs:Instant_2011_02_28_10_56_00 .

sens-obs:MeasureData_AirTemperature_KDAY_2011_02_28_10_56_00
    a      om-owl:MeasureData ;
    om-owl:floatValue "24.1" ;
    om-owl:uom weather:fahrenheit .
   `
sens-obs:Observation_WindSpeed_KDAY_2011_02_28_10_56_00
    a      weather:WindObservation ;
    om-owl:observedProperty
          weather:_WindSpeed ;
    om-owl:procedure sens-obs:System_KDAY ;
    om-owl:result sens-obs:MeasureData_WindSpeed_KDAY_2011_02_28_10_56_00 ;
    om-owl:samplingTime sens-obs:Instant_2011_02_28_10_56_00 .

sens-obs:MeasureData_WindSpeed_KDAY_2011_02_28_10_56_00
    a      om-owl:MeasureData ;
    om-owl:floatValue "19.1" ;
    om-owl:uom weather:milesPerHour .

sens-obs:Observation_Snowfall_KDAY_2011_02_28_10_56_00
    a      weather:PrecipitationObservation ;
    om-owl:observedProperty
          weather:_Snowfall ;
    om-owl:procedure sens-obs:System_KDAY ;
    om-owl:result sens-obs:TruthData_Snowfall_KDAY_2011_02_28_10_56_00 ;
    om-owl:samplingTime sens-obs:Instant_2011_02_28_10_56_00 .

sens-obs:TruthData_Snowfall_KDAY_2011_02_28_10_56_00
    a      om-owl:TruthData ;
    om-owl:booleanValue true ;

sens-obs:Instant_2011_02_28_10_56_00
    a      owl-time:Instant ;
    owl-time:inXSDDateTime
          "2011-02-28T10:56:00"
```

Figure19. Integrated Stream for Sensor System KDAY in RDF

6.3.2 Abstraction of Semantic Sensor Data Streams

The earlier section focused on integrating streams that compose a feature. In this section, we reason over the integrated streams using rules within the weather ontology as background knowledge with the goal of detecting a higher level feature. In order to define a feature, the weather ontology uses the definition for features provided by National Oceanic and Atmospheric

Administration (NOAA)[26]. Figures 20, 21, 22 and 23 gives definitions for Blizzard, Flurry, RainStorm, RainShower features provided by NOAA.

BLIZZARD = High WindSpeed (exceeding 35 mph) AND Snow Precipitation AND Low Visibility (less than ¼ mile), for at minimum 3 hours.

Figure20. Blizzard Definition

FLURRY = Low WindSpeed (below 35 mph) AND Snow Precipitation AND Low Visibility (less than ¼ mile), for at minimum 3 hours.

Figure21. Flurry Definition

RAINSTORM = High WindSpeed (exceeding 35 mph) AND Rain Precipitation AND Temperature (greater than 32F)

Figure22. RainStorm Definition

RAIN SHOWER = Low WindSpeed (below 35 mph) AND Rain Precipitation AND Temperature (greater than 32F)

Figure23. RainShower Definition

As shown in figure 21, detecting a Flurry requires the WindSpeed to be less than 35 miles/hour, temperature below 32F and snow precipitation. Since we have already integrated streams that compose a Flurry in the previous section, here we reason over the stream observation

[26] http://www.noaa.gov/

values using rules to detect a Flurry. This thesis encodes rules within SPARQL query language. The SPAQRL query for all features of interest is then executed over the integrated RDF stream for feature detection. Figure 24 encodes the numerical constraints used to define a Flurry (in figure 21) within SPARQL query language.

```
PREFIX om-owl:<http://knoesis.wright.edu/ssw/ont/sensor-observation.owl#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX sens-obs-streams:<http://knoesis.wright.edu/sensorstreams/>
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX owl-time:<http://www.w3.org/2006/time#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX weather:<http://knoesis.wright.edu/ssw/ont/weather.owl#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX wgs84:<http://www.w3.org/2003/01/geo/wgs84_pos#>

SELECT DISTINCT ?x ?tempValue ?windValue ?time
WHERE{"
 ?x rdf:type sens-obs-streams:System .
 ?x sens-obs-streams:inState OH^^xsd:string .
 ?x om-owl:generatedObservation ?temperature .
 ?temperature om-owl:observedProperty weather:_AirTemperature .
 ?temperature om-owl:samplingTime ?timeInstant .
 ?timeInstant owl-time:inXSDDateTime ?time .
 ?temperature om-owl:result ?tempData .
 ?tempData om-owl:floatValue ?tempValue .
 ?x om-owl:generatedObservation ?rainfall .
 ?rainfall rdf:type weather:WeatherConditionsObservation .
 ?rainfall om-owl:observedProperty weather:_WeatherConditions .
 ?rainfall om-owl:result ?rainData .
 ?rainData om-owl:stringValue ?rain .
 ?x om-owl:generatedObservation ?windSpeed .
 ?windSpeed om-owl:observedProperty weather:_WindSpeed .
 ?windSpeed om-owl:samplingTime ?timeInstant .
 ?timeInstant owl-time:inXSDDateTime ?time .
 ?windSpeed om-owl:result ?windData .
 ?windData om-owl:floatValue ?windValue .
 FILTER(?tempValue > \"32.0\"^^xsd:float)
 FILTER(?windValue > \"35.0\"^^xsd:float)
 FILTER regex(?rain, \"rain\", \"i\")
 }
```

Figure24. Flurry definition encoded in SPARQL

The result of executing the SPARQL rule over integrated RDF stream at time instant t_1 is a feature f_1. The feature f_1 and associated data is then stored as a feature RDF stream. As discussed before, the research within this thesis focuses on detecting and storing higher-level features that humans can comprehend for better decision making. Hence if the rule execution results in no feature, then the lower-level data is discarded. Figure 25 focuses on feature and its relationship within the feature RDF stream. As shown in figure 25, a feature has a "propertyValueProvider" relationship that defines the properties that contribute to the feature. Since a System may detect different features over time, "isDetectedBy" relationship represents the most recent feature (w.r.t time) detected by the system, while "wasDetectedBy" represents the previous feature (w.r.t time) detected by System. The "isBefore" relationship is used to represent the order in which the features are detected by the system. "EventTime" represents the time at which the feature is detected.

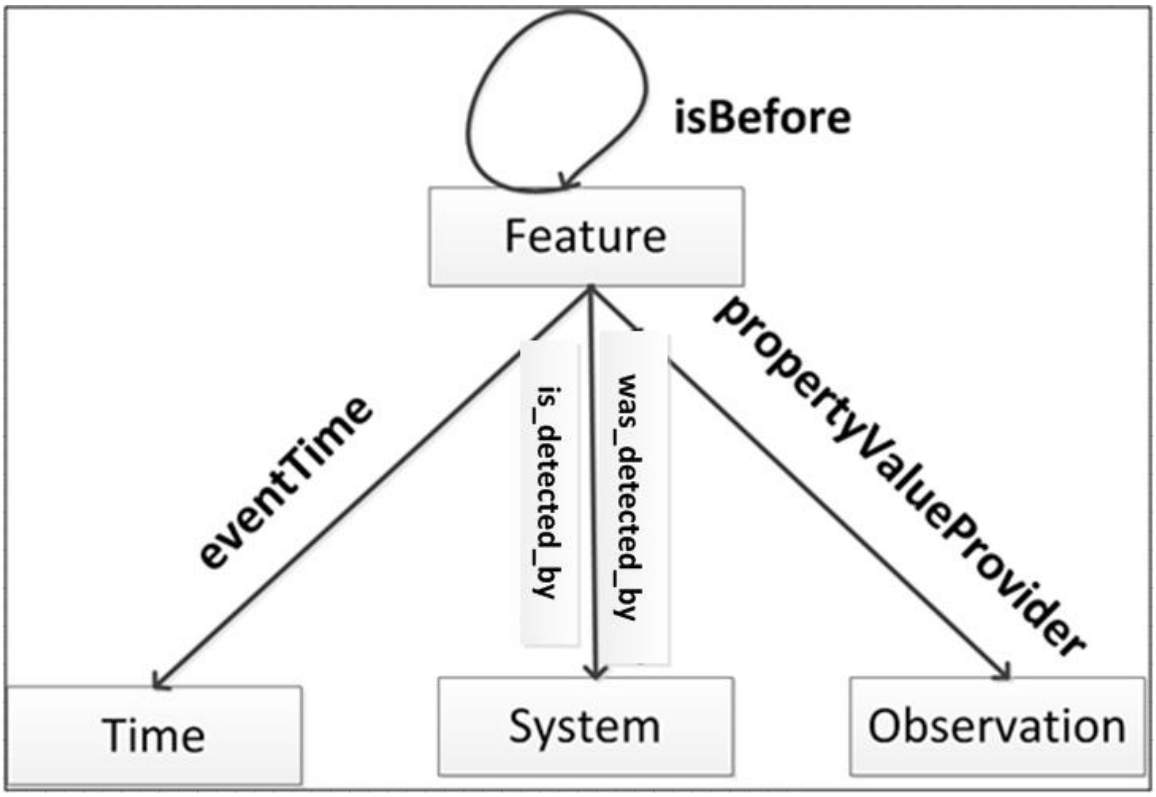


Figure25. Feature RDF stream and its relationship

Figure 26 shows a feature RDF stream generated for system KDAY. The most recent feature detected by system KDAY is a Flurry at time instant "2011-02-28T10:56:00" which is represented using isDetectedBy relation. The isBefore relationship is used to represent the order in which the features were detected. As shown in figure 26, a Flurry feature wasDetectedBy system KDAY at time instant "2011-02-28T10:13:00" which isBefore the feature detected at time instant "2011-02-28T10:56:00".

Figure26. Feature RDF stream

Revisiting the Scenario

Thus summarizing the scenario, part 1 focused on searching for sensors near Dayton James Cox Airport using the spatial context. Part 2 used the sensor detected in part 1 to extract raw sensor streams and convert to RDF stream over the temporal context. Part 3 integrated and reasoned on the RDF streams obtained from part 2 with the goal of searching for a theme (feature). Section 8 shows, creating and storing only meaningful feature results in massive data

reduction. The next section focuses on discussing a very intuitive and easy-to-use interface built to view the feature streams.

VII. Feature Stream Interface

The previous section focused on the infrastructure built for detecting and generating feature streams in real-time. This section is dedicated towards explaining the interface built for viewing the feature streams in real-time. The main idea of this thesis is to create abstractions that humans can easily comprehend and understand. The interface was very carefully built, keeping in mind the ease of understanding aspect for humans. Figure 28 shows a snapshot of the interface built. The interface would be explained in three steps for ease of understanding.

7.1 Begin Search: The interface provides two options to begin search, marked in figure 27. The options are given below:

Figure27. Feature Streams Interface (Begin Search)

7.1.1 Search Bar: The search bar would allow a user to search for stream of features in a specific named location like "Dayton James Cox Airport". In this case the result would contain the sensors near the location.

7.1.2 State Selection: The user can search for stream of features for an entire state. This is done by clicking on the specific state. In this case the result would contain all the sensors in the state that are currently active. Figure 29 shows the result of selecting Ohio as the state. The result contains all the sensors in Ohio that are currently active. The snapshot contains few sensors to provide an uncluttered view, however in reality Ohio contains greater than ~100 sensors.

7.2 Feature Selection: In this step the user is given an option to choose the features of interest. A user can choose the features he/she is interested in by adding the features to the Event Bag using a drag and drop option. Figure 28 shows Flurry, RainStorm and RainShower added to the event bag as features of interest. On submit the interface would provide only the sensors that are currently detecting the features of interest. In this case, it would provide all sensors that are currently

detecting RainStorm, RainShower or Flurry. For ease, we have provided a unique icon for each feature detected.

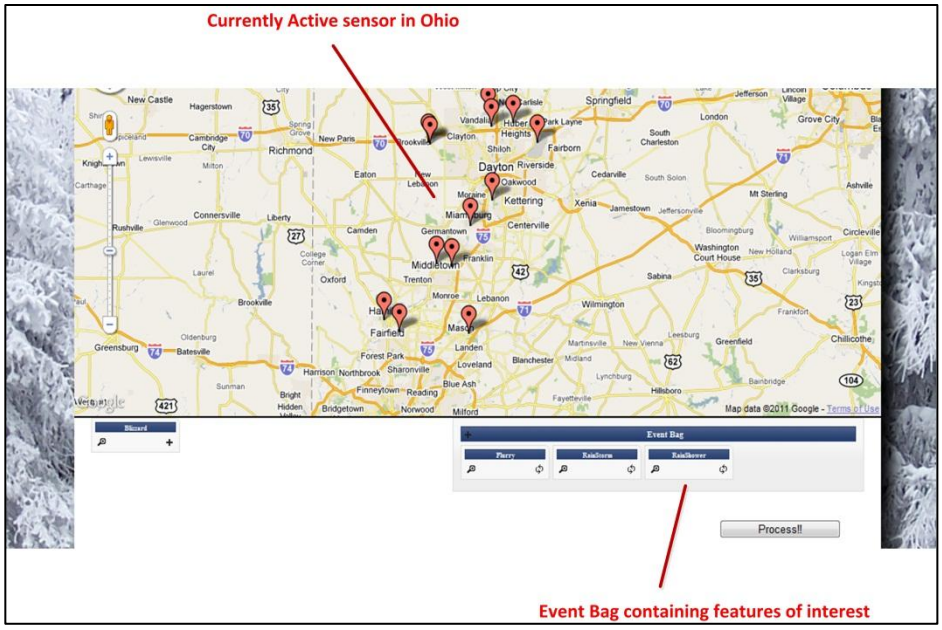


Figure28. Feature Streams Interface (Feature Selection)

7.3 Feature Streams View: Step 7.2 will provide a list of sensors (represented by feature icons) currently detecting a feature of interest (RainShower, RainStorm, Flurry in this case). In order to view the stream of features, the user would have to select a specific sensor of interest. Figure 29 provides a snapshot of the stream of features detected by sensor system KHAO. The last feature detected by sensor system KHAO is RainShower and hence is represented by a RainShower icon. We provide a graphical view, to observe the stream of features that were detected by sensor system KHAO and values that contributed towards the features over time. Figure 29

shows 3 graphs (Temperature, WindSpeed and Precipitation) representing the lower-level data values and Feature graph that represent the features detected. The four graphs are aligned over time to represent the combination of lower-level values that contribute in feature detection.

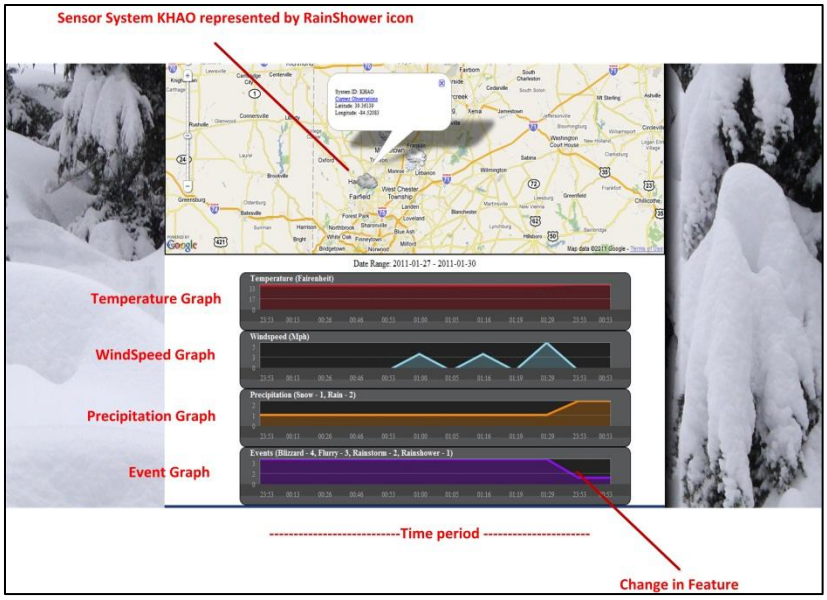


Figure29. Feature Streams Interface (Feature Streams View)

VIII. EVALUATION

The previous section discussed the interface built for viewing real-time feature streams. This section evaluates the approach with respect to amount of storage required to store the abstractions generated. As shown in figure 30, with the emergence of dynamic information sources, total amount of information generated has completely surpassed the total storage capacity.

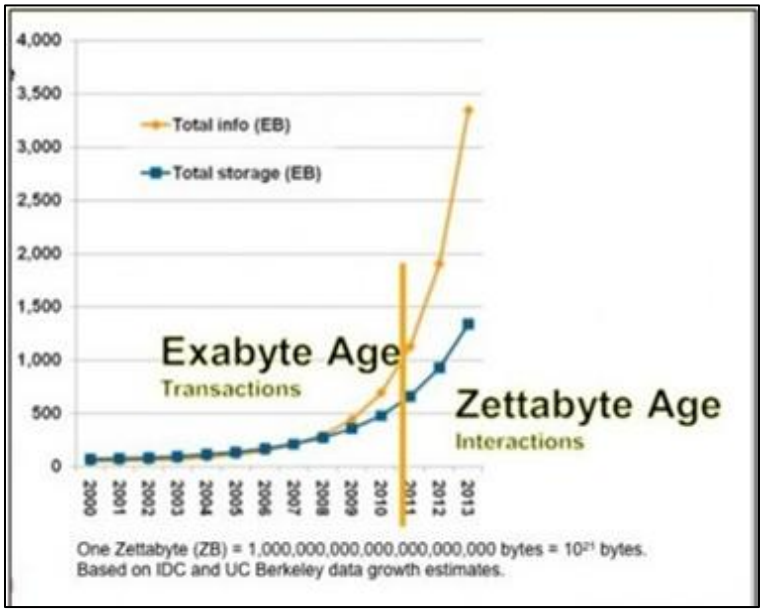


Figure30. Rate of Information Generation

With this data explosion, real-time analysis and storing only the relevant abstractions (meaningful summaries of large amount of data) has become extremely important. This evaluation shows the amount of data reduced by storing only the meaningful abstractions and the associated cases.

Since we are interested in detecting features such as Blizzard, Flurry, RainStorm and RainShower, the experiment requires raw data that contains these features. With this view, the storm that occurred in Nevada between April 1st and April 6th 2003 was used. The data during this period for all the sensors in Nevada and the neighboring states was collected using MesoWest API. This data is also a part of the Linked Observation dataset added to the Linked Open Data Cloud. [18] As stated before, this is the first dataset to add sensor observations to the Linked Open Data Cloud. The dataset contains a total of 111,456 observations. The results for data reduction can be found in the figure 31 below.

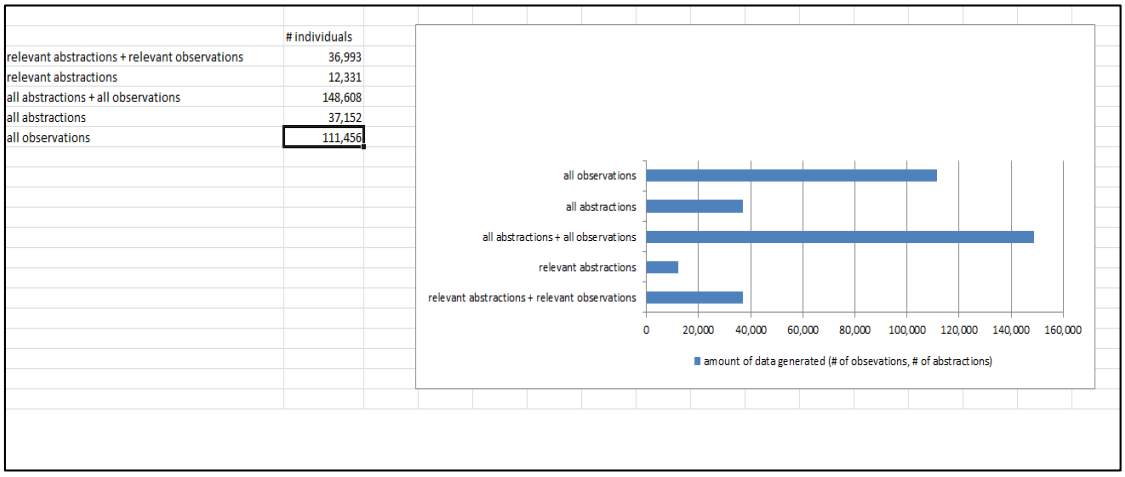


Figure31. Storage Evaluation Results

The graph in the figure shows the specific case on the Y axis and the total number of observations stored on the X axis. As shown in the figure there is 89% decrease in the amount of

data stored, if only the relevant abstractions are stored. The relevant abstractions are Blizzard, Flurry, RainStorm and RainShower. This is the amount of data reduction when an event actually occurred. However over a year, on an average there are not more than 4-5 events which mean the data reduction would be massive if only the abstractions were stored. Also practically from a human perspective, one is interested in a sequence of features being observed and not when the weather is clear (no features detected). Sometimes applications require the data associated with the abstractions (relevant abstractions + relevant observations) which still results in 67% reduction in the amount of data stored. Some weather applications require "Clear" to be stored in cases where no feature is being detected. The term "all abstractions" is used to define this case which again leads to 67% data reduction. Hence summarizing this section, storing of relevant abstractions would help solving the problem of data explosion. Also it is easy to get a feel of the lower-level conditions by looking at a feature stored. However, the reverse is much difficult especially with the scale of data.

IX. Conclusion and Future Work

Today, real-time data generated by various dynamic information sources is available on the web. Huge number systems and algorithms are built for on-the-fly processing of real-time data. However the challenge of making sense of this huge amount of data to recognize features in near real-time that humans care about still remains.

This thesis focused on solving this challenging problem. An infrastructure was built for integrating and analyzing real-time lower-level data streams and generating feature streams in real-time. This work also provides an intuitive interface to view these feature streams making it easy for humans to analyze and make better decisions.

This work focuses on analysis of machine sensor data. An extension of this work could be integrating this platform with social networking platforms like Twitris [50] that deal with social sensing data. Twitris is a Semantic Web application that facilitates understanding of social perceptions by spatio-temporal-thematic processing of massive amounts of event-centric data. Twitris also covers context based semantic integration of multiple Web resources and exposes semantically enriched social data to the public domain. Integrating social sensing and machine

sensing data would provide a more descriptive picture of real-world events.

References

[1]. Gigaom Atricle on Big Data, 2010, http://gigaom.com/cloud/sensor-networks-top-social-networks-for-big-data-2/

[2]. Software must adapt or Die, 2010, http://www.readwriteweb.com/archives/data_analytics_software_must_adapt.php

[3]. Samet, R. and Tural, S., "WEB Based Real-Time Meteorological Data Analyses and Mapping Information System", Proceedings of the 11th WSEAS International Conference on Automation & Information (ICAI'10), Recent Advances in Automation & Information " G. Enescu" University, Iasi, Romania, June 13-15, pp. 80-85, 2010

[4]. MesoWest, 2011, http://mesowest.utah.edu/

[5]. Next Generation Weather Lab, 2011, http://weather.cod.edu/analysis/

[6]. Wikipedia Web 1.0, 2011, http://en.wikipedia.org/wiki/Web_1.0

[7]. Wikipedia Web 2.0, 2011, http://en.wikipedia.org/wiki/Web_2.0

[8]. Sensor Web Enablement, 2011, http://www.opengeospatial.org/projects/groups/sensorweb

[9]. W3C, Sensor Wikipedia, 2011, http://www.w3.org/2001/sw/wiki/Main_Page

[10]. Tim Berners-Lee, 2011, http://norman.walsh.name/knows/who/tim-berners-lee.html

[11]. Resource Description Framework Wikipedia, 2011, http://en.wikipedia.org/wiki/Resource_Description_Framework

[12]. Linked Data Design Issues, 2010, Design Issues: Linked Data, http://www.w3.org/DesignIssues/LinkedData.html

[13]. Linked Open Data Cloud, 2011, http://linkeddata.org/

[14]. RDF Syntax Specification,2011, http://www.w3.org/TR/REC-rdf-syntax/

[15]. Kno.e.sis Sensor Web Group,2011, http://semantic-sensor-web.com/

[16]. Kno.e.sis Webpage, 2011, http://knoesis.wright.edu/

[17]. Wikipedia Ontology Computer Science, 2011, http://en.wikipedia.org/wiki/Ontology_(information_science)

[18]. Kno.e.sis LinkedSensorData, 2011, http://wiki.knoesis.org/index.php/LinkedSensorData

[19]. Amit Sheth and Matthew Perry, 'Traveling the Semantic Web through Space, Time and Theme,' IEEE Internet Computing, 12, (no.2), February/March 2008, pp.81-86

[20]. Joshua Pschorr, Cory Henson, Harshal Patni, and Amit P. Sheth, 'Sensor Discovery on Linked Data', Kno.e.sis Center Technical Report 2010

[21]. Geonames, 2011, http://www.geonames.org

[22]. NASA SWEET ontology, 2011, http://sweet.jpl.nasa.gov/

[23]. Google Weather API,2011, http://blog.programmableweb.com/2010/02/08/googles-secret-weather-api/

[24]. NOAA Weather Service, 2011, http://www.programmableweb.com/api/noaa-weather-service

[25]. D. Terry, D. Goldberg, D. Nichols, and B. Oki. Continuous queries over append-only databases. In Proc. of the 1992 ACM SIGMOD Intl. Conf. on Management of Data, pages 321–330, June 1992.

[26]. Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. 2007. Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[27]. Wikipedia Data Stream Management Systems, 2011, http://en.wikipedia.org/wiki/Data_Stream_Management_System

[28]. Benjamin Nguyen, Serge Abiteboul, gory Cobena, and Miha; Preda. 2001. Monitoring XML data on the Web. In Proceedings of the 2001 ACM SIGMOD international conference on Management of data (SIGMOD '01), Timos Sellis (Ed.). ACM, New York, NY, USA, 437-448. DOI=10.1145/375663.375723 http://doi.acm.org/10CQ5/375663.375723

[29]. L. Liu, C. Pu, and W. Tang. Continual queries for internet scale event-driven information delivery. IEEE Trans. on Knowledge and Data Engineering, 11(4):583–590, Aug. 1999.

[30]. J. Chen, D. J. DeWitt, F. Tian, and Y. Wang. NiagraCQ: A scalable continuous query system for internet databases. In Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data, pages 379–390, May 2000.

[31]. Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and issues in data stream systems. In Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '02). ACM, New York, NY, USA, 1-16. DOI=10.1145/543613.543615 http://doi.acm.org/10.1145/543613.543615

[32]. D. Carney, U. Cetinternel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams – a new class of dbms applications. Technical Report CS-02-01, Department of Computer Science, Brown University, Feb. 2002.

[33]. R. Avnur and J. Hellerstein. Eddies: Continuously adaptive query processing. In Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data, pages 261–272, May 2000.

[34]. J. Hellerstein, M. Franklin, et al. Adaptive query processing: Technology in evolution. IEEE Data Engineering Bulletin, 23(2):7–18, June 2000.

[35]. S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In Proc. of the 2002 Intl. Conf. on Data Engineering, Feb. 2002.

[36]. S. Madden, J. Hellerstein, M. Shah, and V. Raman. Continuously adaptive continuous queries over streams. In Proc. of the 2002 ACM SIGMOD Intl. Conf. on Management of Data, June 2002.

[37]. P. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In Proc. Of the 2001 ACM Symp. on Parallel Algorithms and Architectures, pages 281–291, 2001

[38]. Davide Francesco Barbieri, Daniele Braga, Stefano Ceri, Emanuele Della Valle, and Michael Grossniklaus. 2009. C-SPARQL: SPARQL for continuous querying. In Proceedings of the 18th international conference on World wide web (WWW '09). ACM, New York, NY, USA, 1061-1062. DOI=10.1145/1526709.1526856 http://doi.acm.org/10.1145/1526709.1526856

[39]. A. Bolles, M. Grawunder, and J. Jacobi. Streaming SPARQL - extending SPARQL to process data streams. In Proceedings of the 5th European Semantic Web Conference (ESWC'08), pages 448{462, 2008.

[40]. Darko Anicic, Paul Fodor, Sebastian Rudolph, and Nenad Stojanovic. 2011. EP-SPARQL: a unified language for event processing and stream reasoning. In Proceedings of the 20th international conference on World wide web (WWW '11). ACM, New York, NY, USA, 635-644. DOI=10.1145/1963405.1963495 http://doi.acm.org/10.1145/1963405.1963495

[41]. J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Effecient pattern matching over event streams. In Proceedings of the 28th ACM SIGMOD Conference, pages 147-160, 2008.

[42]. S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah. Telegraphcq: Continuous dataflow processing for an uncertain world. In Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR'03), 2003.

[43]. M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S. B. Zdonik. Scalable distributed stream processing. In Proceedings of the 1st Biennial Conference on Innovative Data Systems Research (CIDR'03), 2003.

[44]. O. Walavalkar, A. Joshi, T. Finin, and Y. Yesha. Streaming knowledge bases. In International Workshop on Scalable Semantic Web Knowledge Base Systems, 2008

[45]. C. Gutierrez, C. A. Hurtado, and A. A. Vaisman. Introducing time into rdf. The IEEE Transactions on Knowledge and Data Engineering, 19(2):207-218, 2007.

[46]. M. Perry, A. P. Sheth, and P. Jain. SPARQLST: Extending SPARQL to support spatiotemporal queries. In Technical Report. KNOESIS-TR-2009-01, 2008

[47]. M. Koubarakis and K. Kyzirakos. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. In Proceedings of the 7th Extended Semantic Web Conference (ESWC'10), pages 425-439, 2010.

[48]. F. Grandi. T-SPARQL: a TSQL2-like temporal query language for RDF. In International Workshop on on Querying Graph Structured Data, pages 21-30, 2010.

[49]. Harshal Patni, Cory Henson, Amit Sheth, 'Linked Sensor Data,' In: Proceedings of 2010 International Symposium on Collaborative Technologies and Systems (CTS 2010), Chicago, IL, May 17-21, 2010

[50]. Twitris, 2011, Twitris, http://twitris.knoesis.org

[51]. Lefort, L., Henson, C., Taylor, K., Barnaghi, P., Compton, M., Corcho, O., Garcia-Castro, R., Graybeal, J., Herzog, A., Janowicz, K., Neuhaus, H., Nikolov, A., and Page, K.: Semantic Sensor Network XG Final Report, W3C Incubator Group Report (2011).

[52]. H. Neuhaus and M. Compton *The Semantic Sensor Network Ontology: A Generic Language to Describe Sensor Assets*. In AGILE Workshop Challenges in Geospatial Data Harmonisation, 2009

[53]. SSN-XG, http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/#xlink:DeRose:REC:2001

[54]. Danh Le-Phuoc, Hoan Nguyen Mau Quoc, Josiane Xavier Parreira, and Manfred Hauswirth, "The Linked Sensor Middleware - Connecting the real

[55]. world and the Semantic Web", In Proceedings of the 10th International Semantic Web Conference (ISWC2011), Springer 2011.